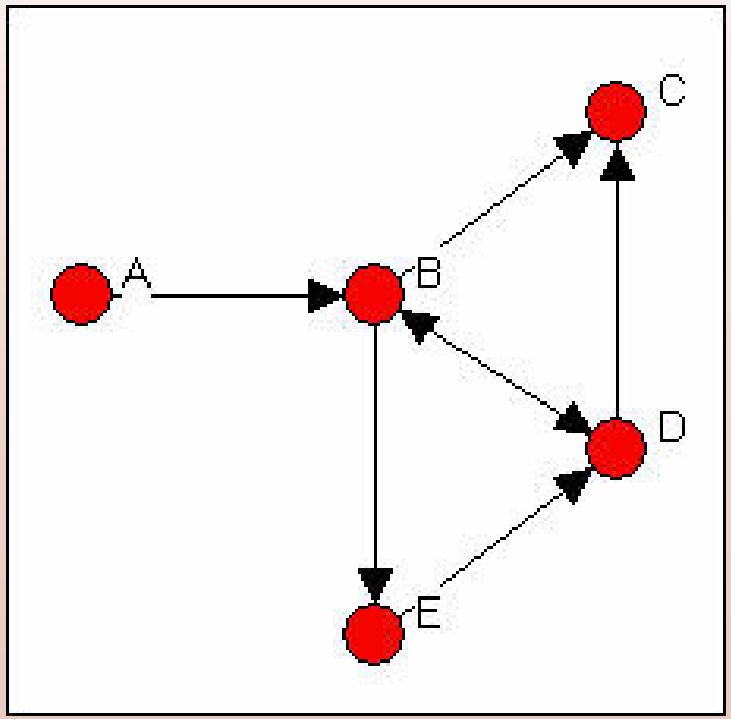
CMSC 250
Discrete Structures

Graphs and Trees

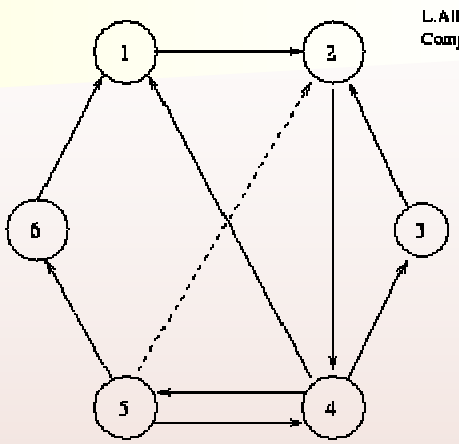
Graphs

- Vertices
- Edges (endpoints)

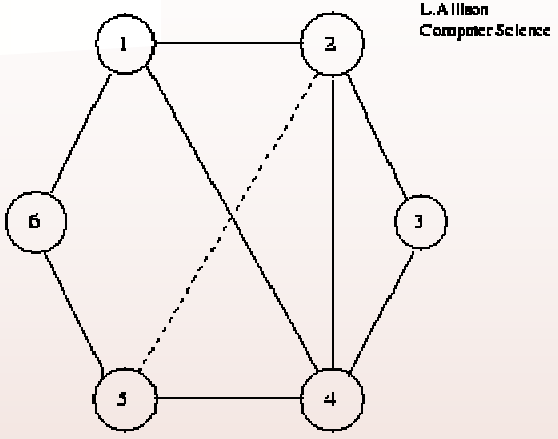


Types of Graphs

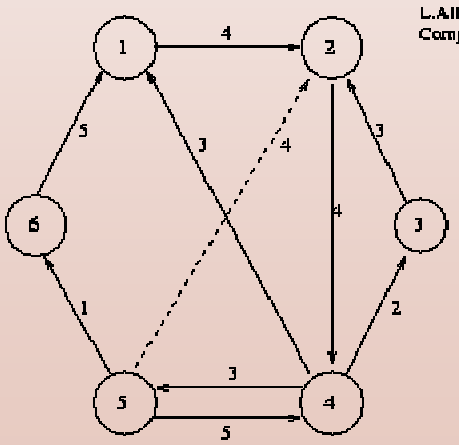
- Directed – order counts when discussing edges
- Undirected (bidirectional)
- Weighted – each edge has a value associated with it
- Unweighted



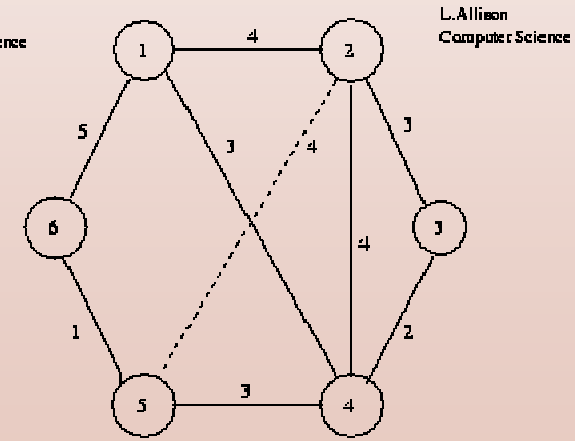
Directed Graph



Undirected Graph

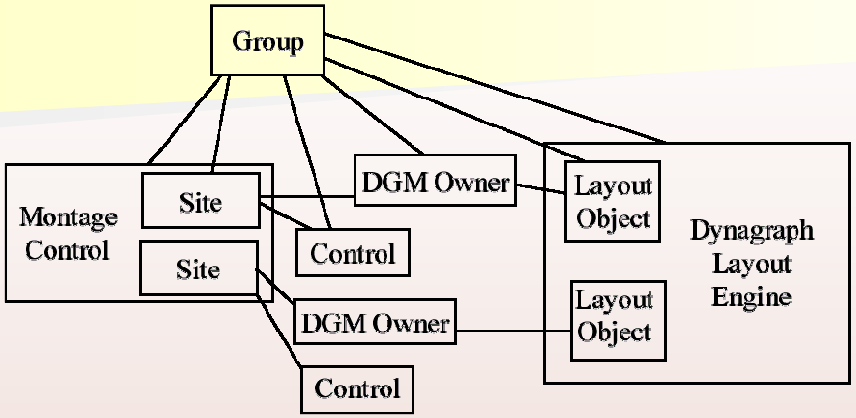


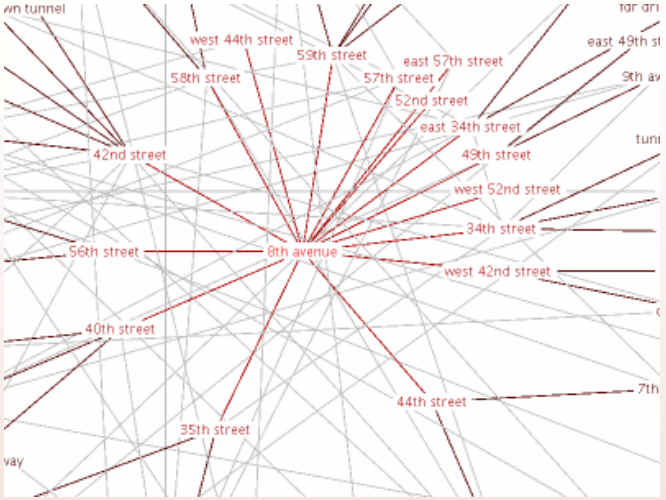
Weighted Directed Graph



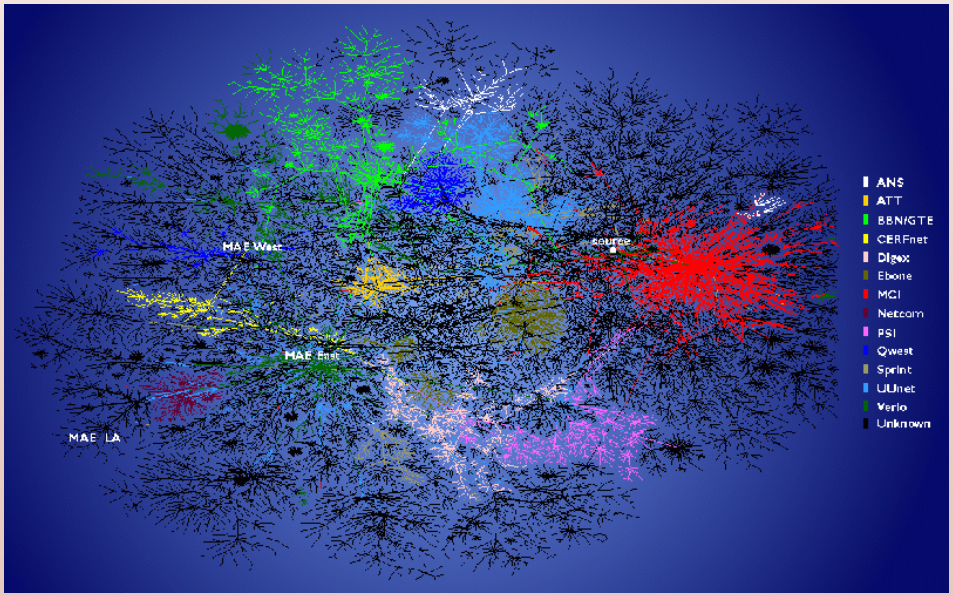
Weighted Undirected Graph

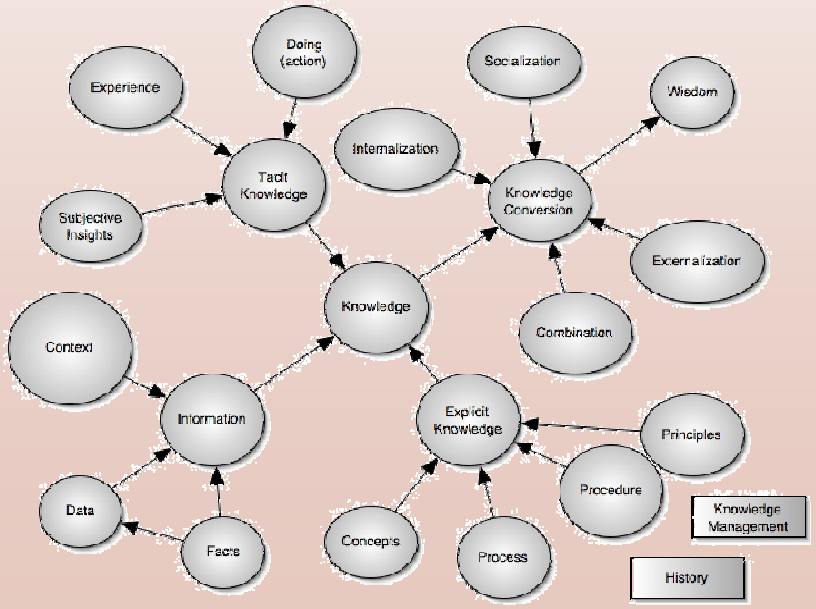
Examples





http://richard.jones.name/google-hacks/google-cartography/google-cartography.html



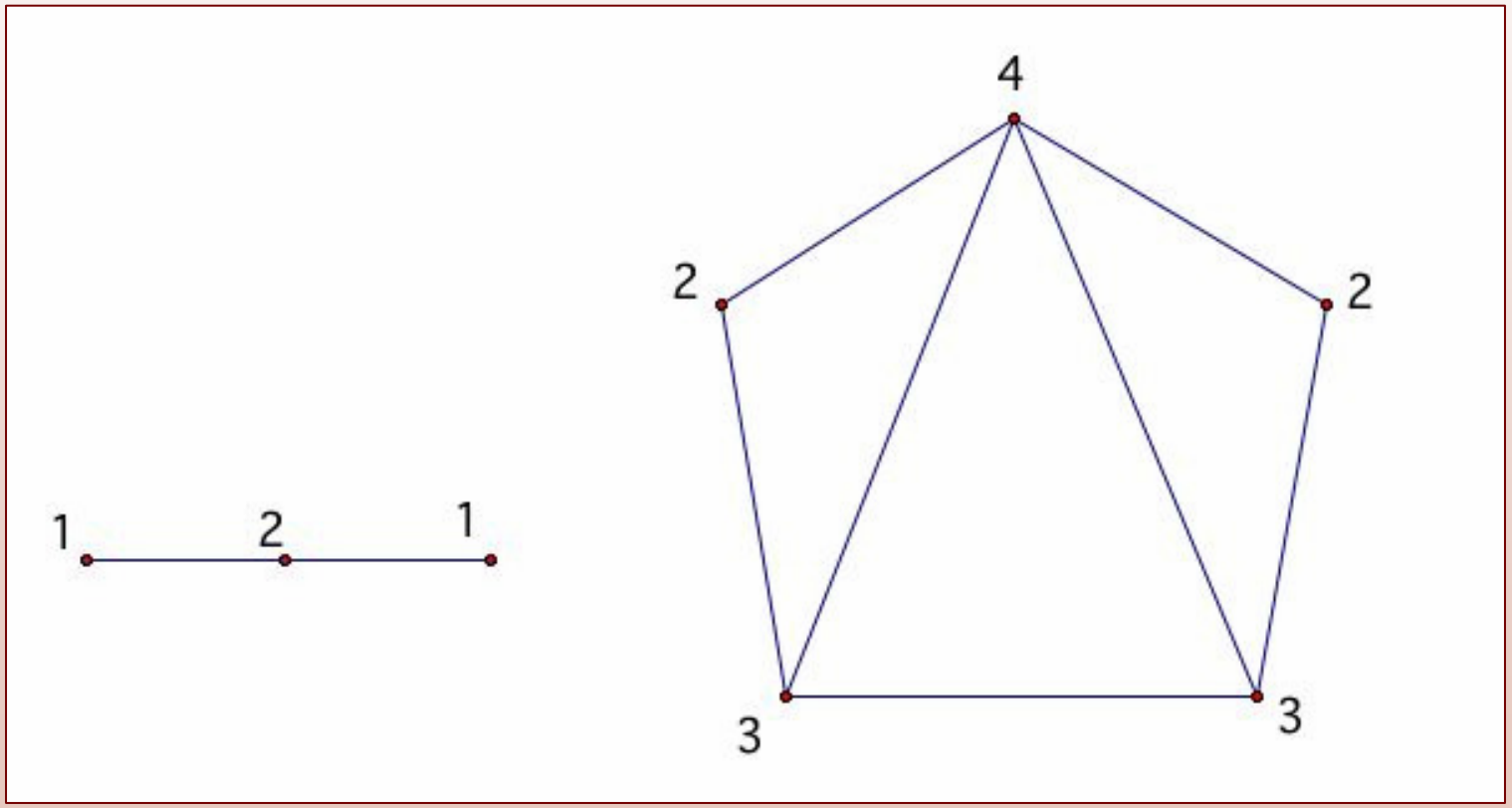


Special Graphs

- Simple – does not have any loops or parallel edges
- Complete graphs – there is an edge "between" every possible tuple of vertices
- Bipartite graph – V can be partitioned into V1 and V2, such that:
  - $(x,y) \in E \rightarrow (x \in V1 \wedge y \in V2) \vee (x \in V2 \wedge y \in V1)$
- Sub graphs
  - G1 is a subset of G2 iff
    - Every vertex in G1 is in G2
    - Every edge in G1 is in G2
- Connected graph – can get from any vertex to another via edges in the graph

Degree of Vertex

- Defined as the number of edges attached to the vertex



Handshake Theorem

- If G is any graph, then the sum of the degrees of all the vertices of G equals twice the number of edges of G.
- Specifically, if the vertices of G are v_1, v_2, …, v_n, where n is a nonnegative integer, then:
  - The total degree of G = $d(v_1)+d(v_2)+\ldots+d(v_n)$
    = $2 \cdot$ (the number of edges of G)

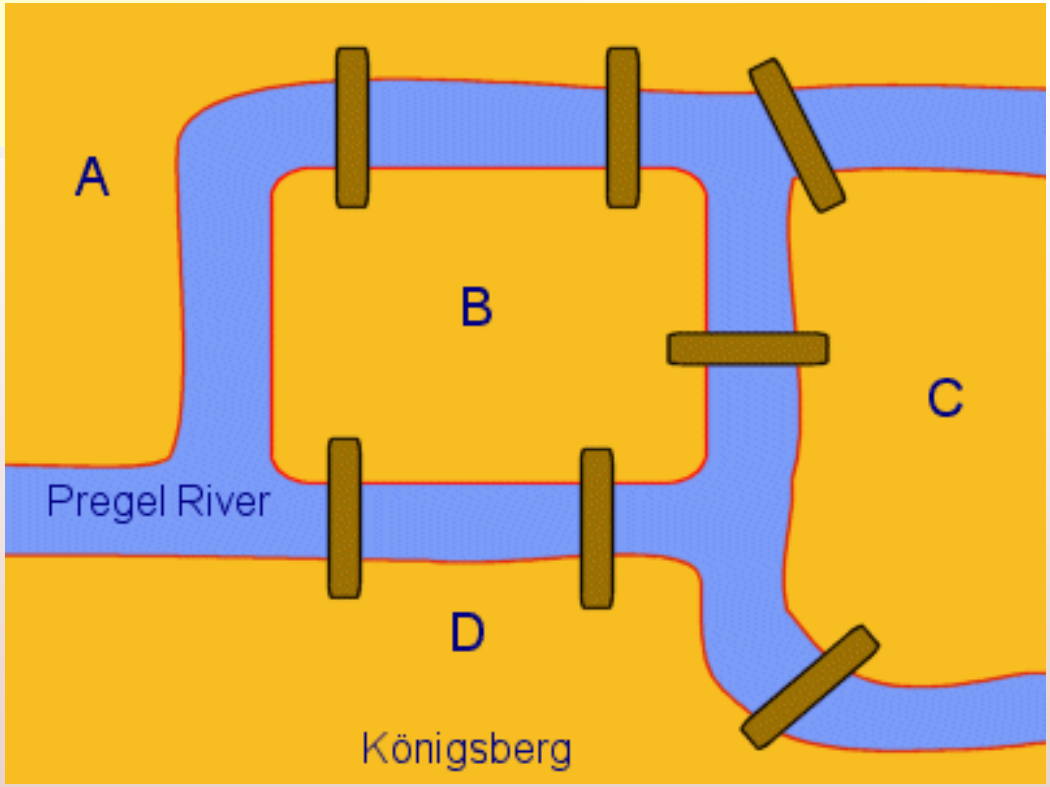
Prove: Sum of all degrees is even

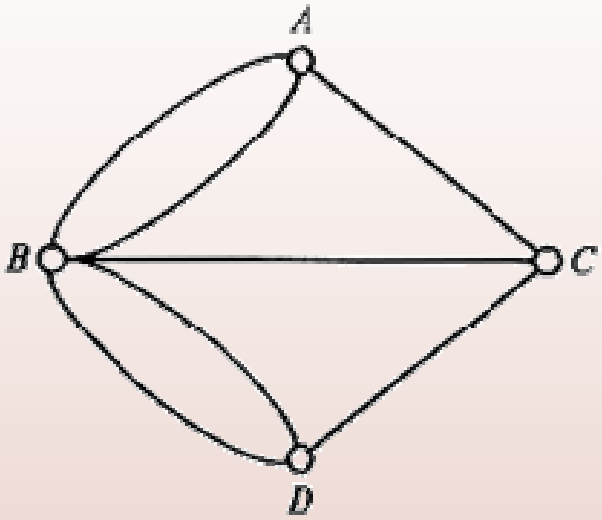
- Prove that the sum of the degrees of all vertices in a graph is even.

Prove: Even # vertices w/ odd degree

- In any graph, there are an even number of vertices with odd degree

Seven Bridges of Königsberg





Is it possible for a person to take a walk around town, starting and ending at the same location and crossing each of the seven bridges exactly once?

No

Definitions

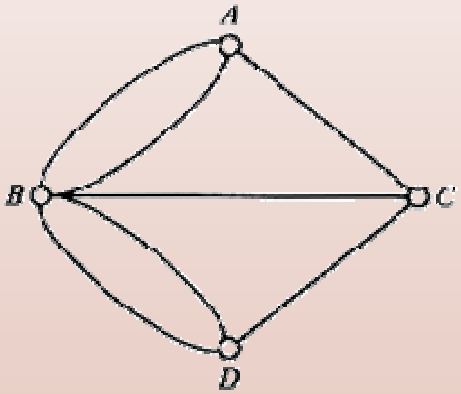
- Walk from two vertices alternating sequence of adjacent vertices and edges
  - Trivial walk from v to v consists of single vertex
- Path – does not contain a repeated edge
- Simple path – does not contain a repeated vertex
- Closed walk – starts and ends at same vertex
- Circuit – a closed walk without repeated edge
- Simple circuit – no repeated vertex except first and last
- Connectedness – if a walk from one to the other

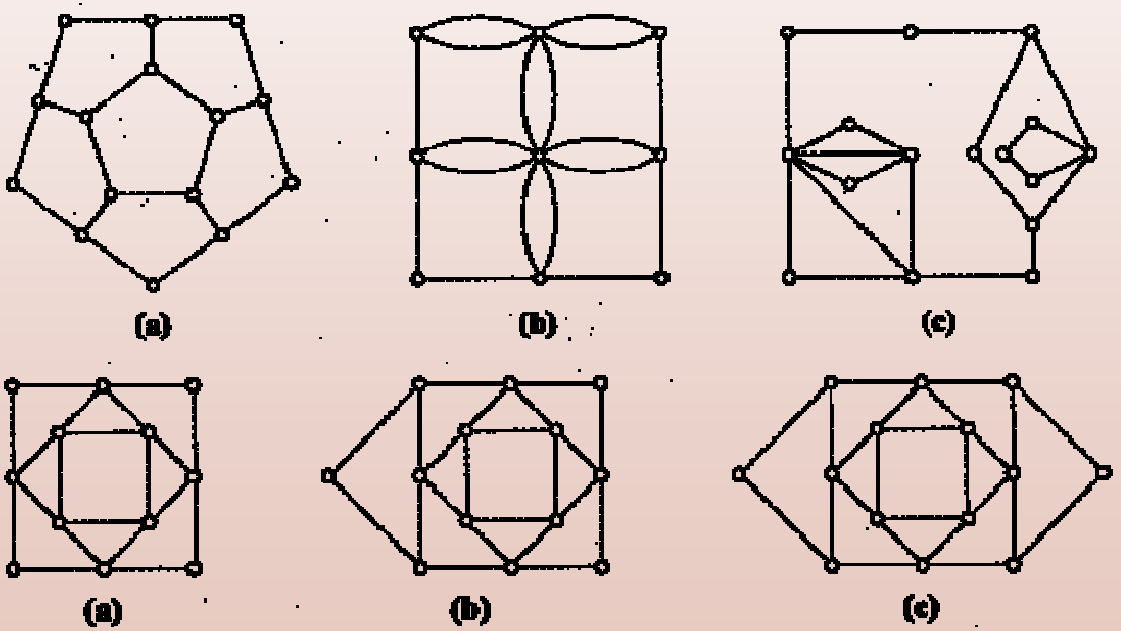
Euler Circuits

- A circuit that contains every vertex and every edge of G.
- A sequence of adjacent vertices and edges
  - That starts and ends at the same vertex,
  - uses every vertex of G at least once, and
  - uses every edge of G exactly once.

If a graph has an Euler circuit, every vertex has even degree.

- Contrapositive: if some vertex has odd degree, then the graph does not have an Euler circuit.





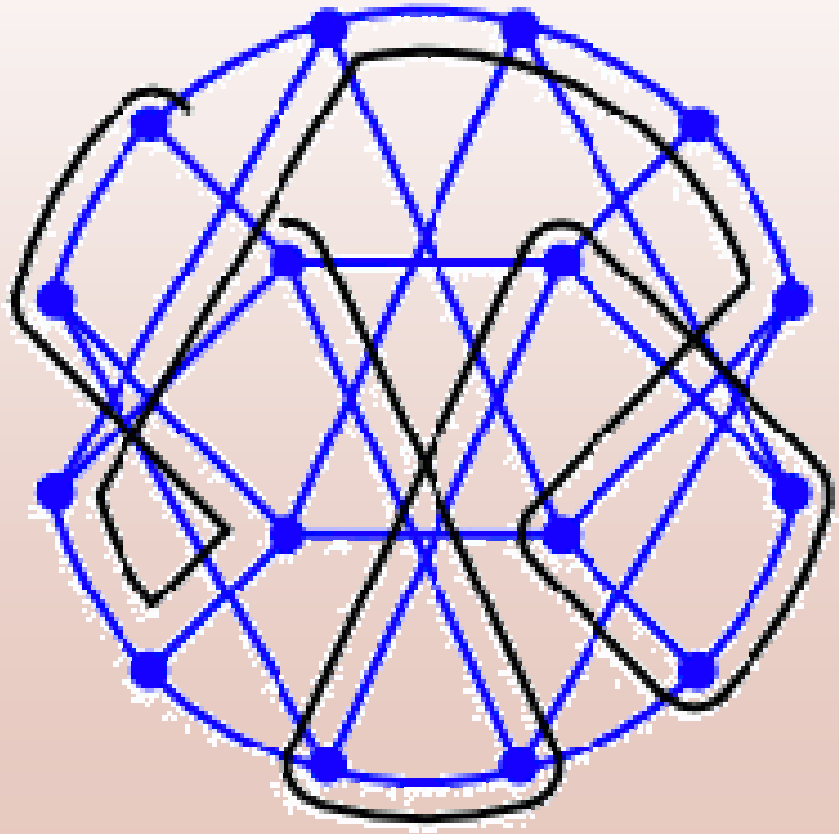
If every vertex of nonempty graph has even degree and if graph is connected, then the graph has an Euler circuit.

Euler Circuit Proofs

- If every vertex of nonempty graph has even degree and if graph is connected, then the graph has an Euler circuit.
- A graph G has an Euler circuit if, and only if, G is connected and every vertex of G has even degree.

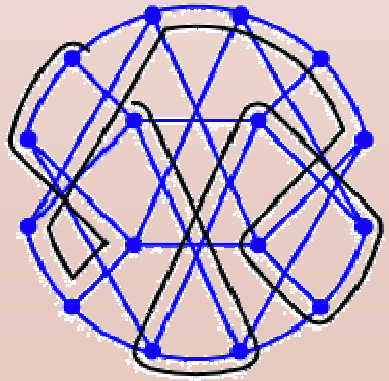
Hamiltonian Path

A path in an undirected graph which visits each vertex exactly once.



Hamiltonian Circuit

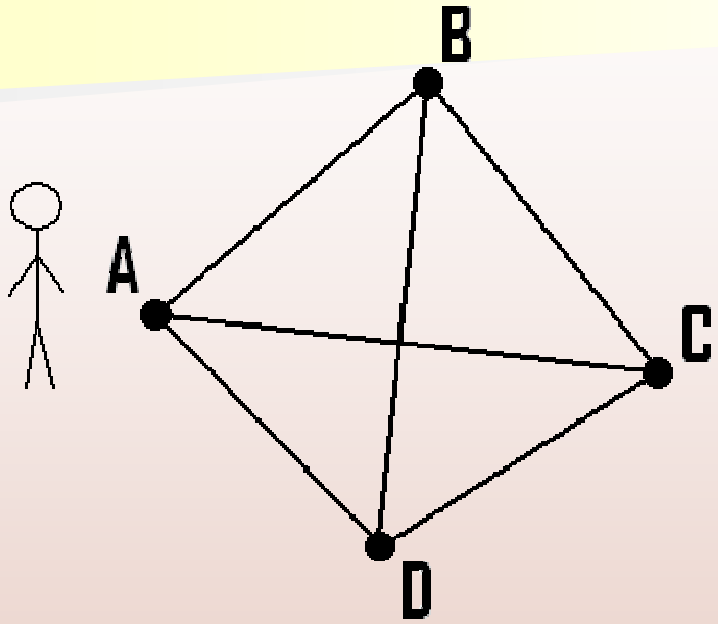
- A simple circuit that includes every vertex of G.
- A sequence of adjacent vertices and distinct edges in which every vertex of G appears exactly once, except for the first and last, which are the same.

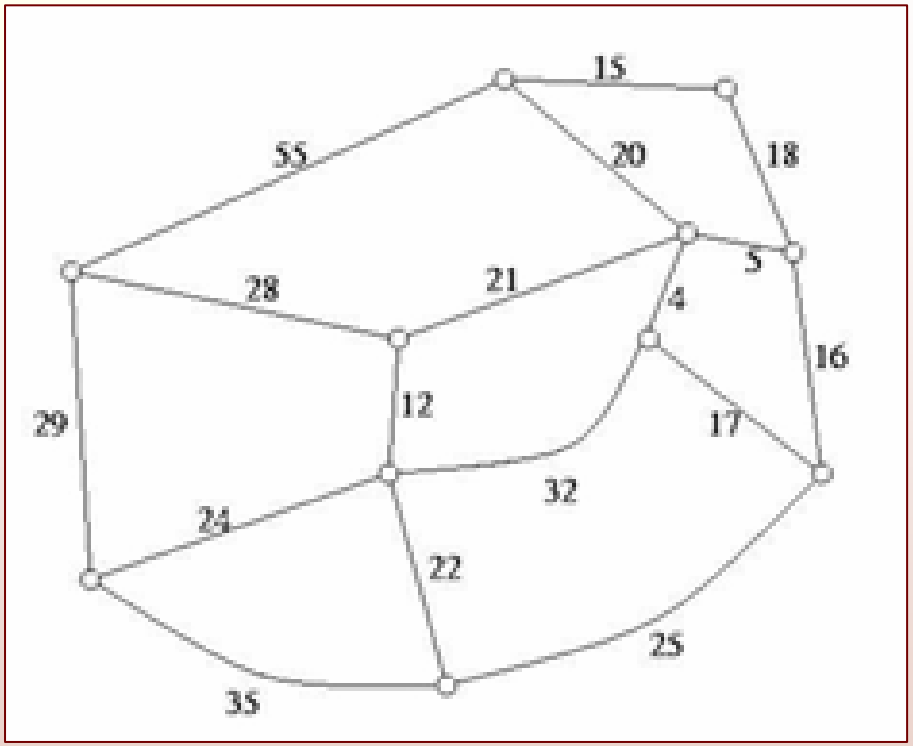


Hamiltonian Circuit

- Proved simple criterion for determining whether a graph has an Euler circuit
- No analogous criterion for determining whether a graph has a Hamiltonian circuit
- Nor is there an efficient algorithm for finding such an algorithm

Traveling Salesman Problem



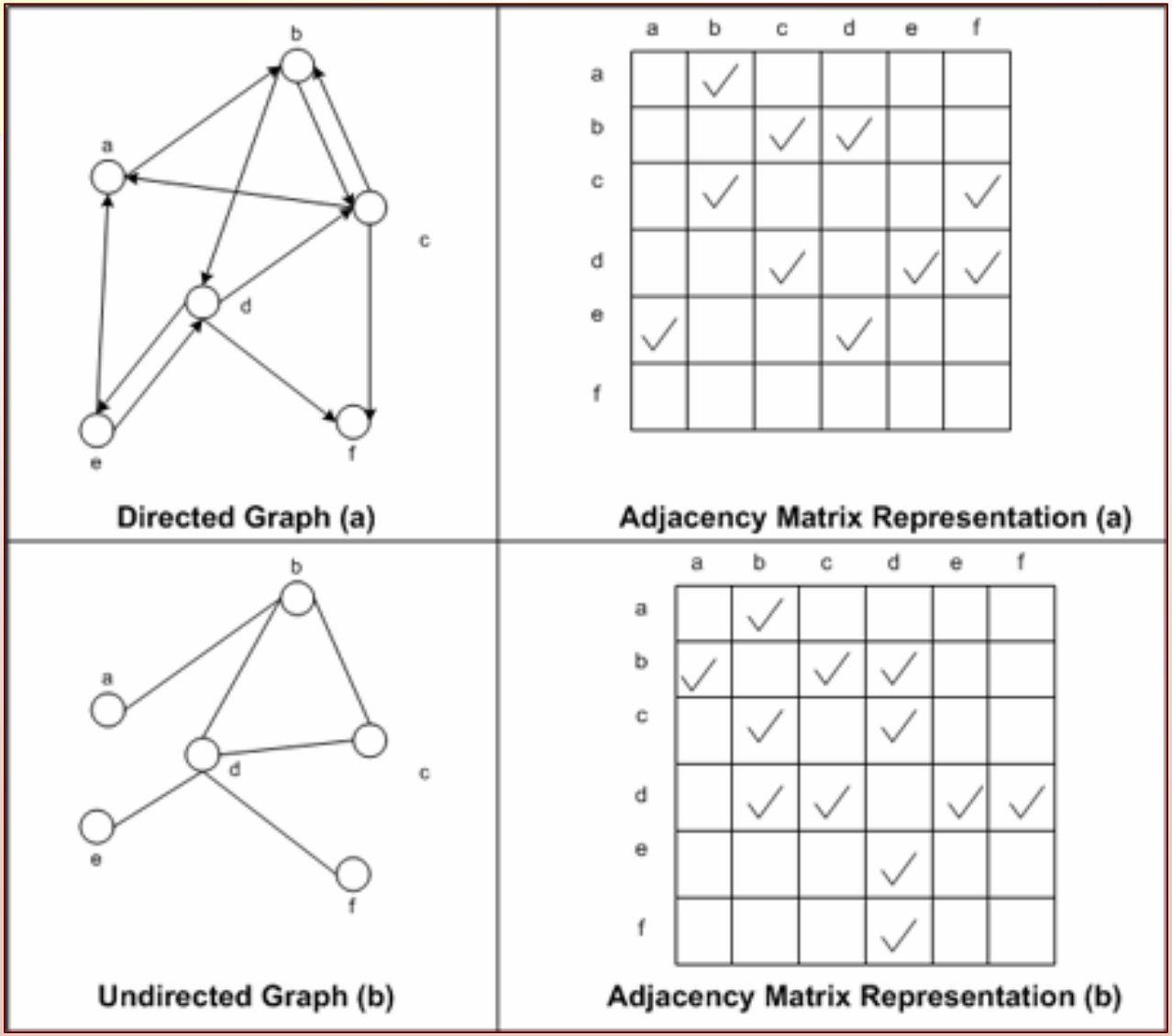


- http://en.wikipedia.org/wiki/Traveling_Salesman_Problem

TSP

- One way to solve the general problem is to:
  - Write down all Hamiltonian circuits
  - Compute total distance for each
  - Pick one for which total is minimal
- What if graph has 30 vertices:
  - 29! =8.84 x 10^{30} different Hamiltonian circuits
  - If each circuit could be found and total distance computed in a nanosecond, then would take:
    - 2.8 x 10^{14} years!!!
  - No known algorithm that is more efficient!!!
  - Some that find "pretty good" solutions

Matrix Representations of Graphs

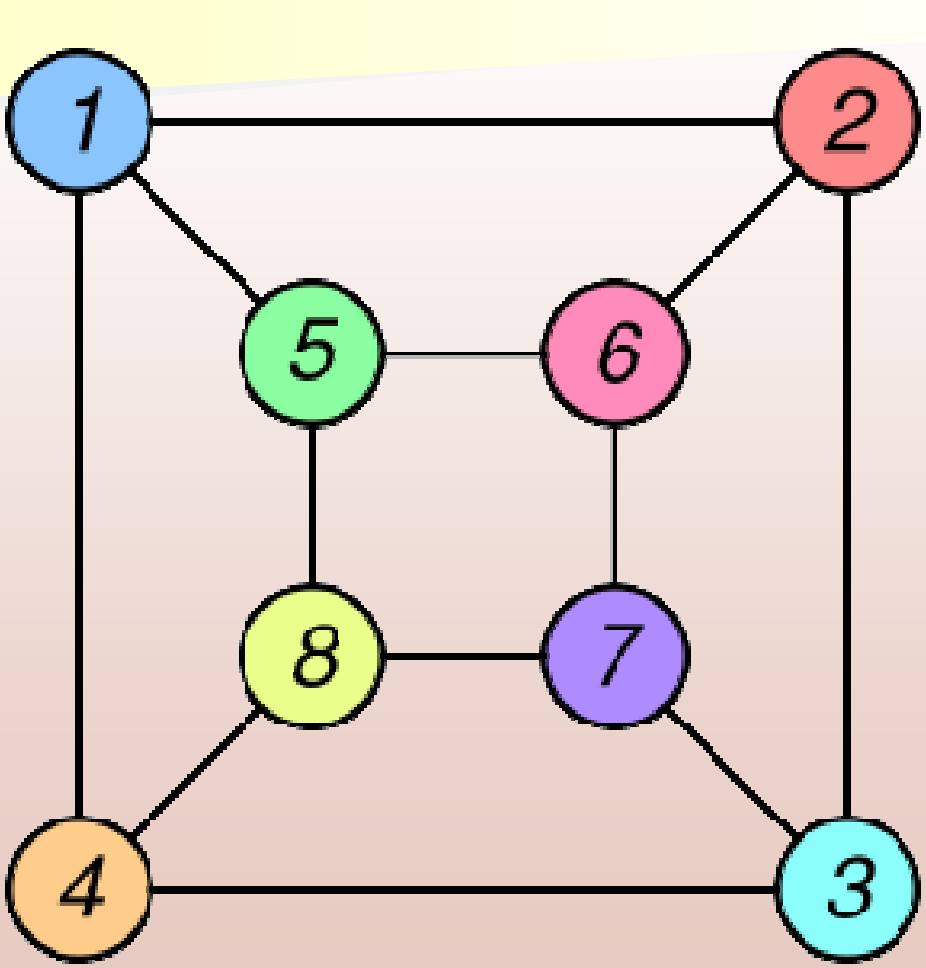


Matrices and Connected Components

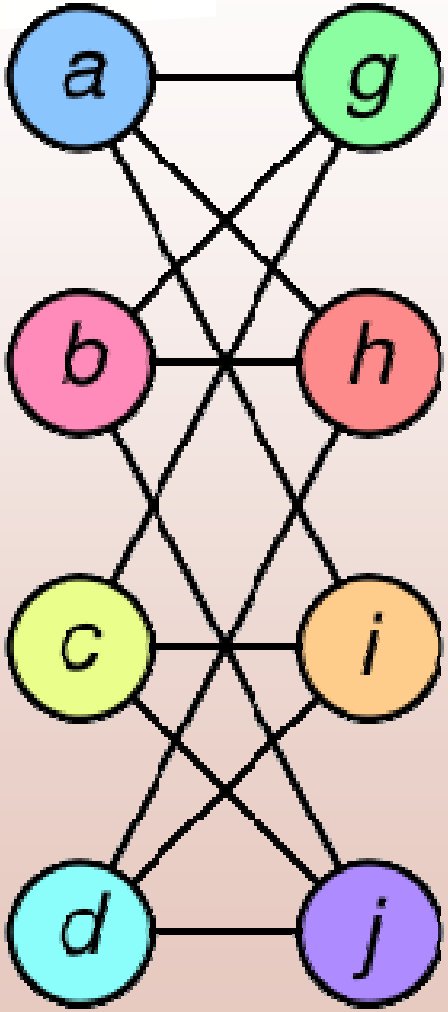
Counting Walks of Length n

- Matrix multiplication

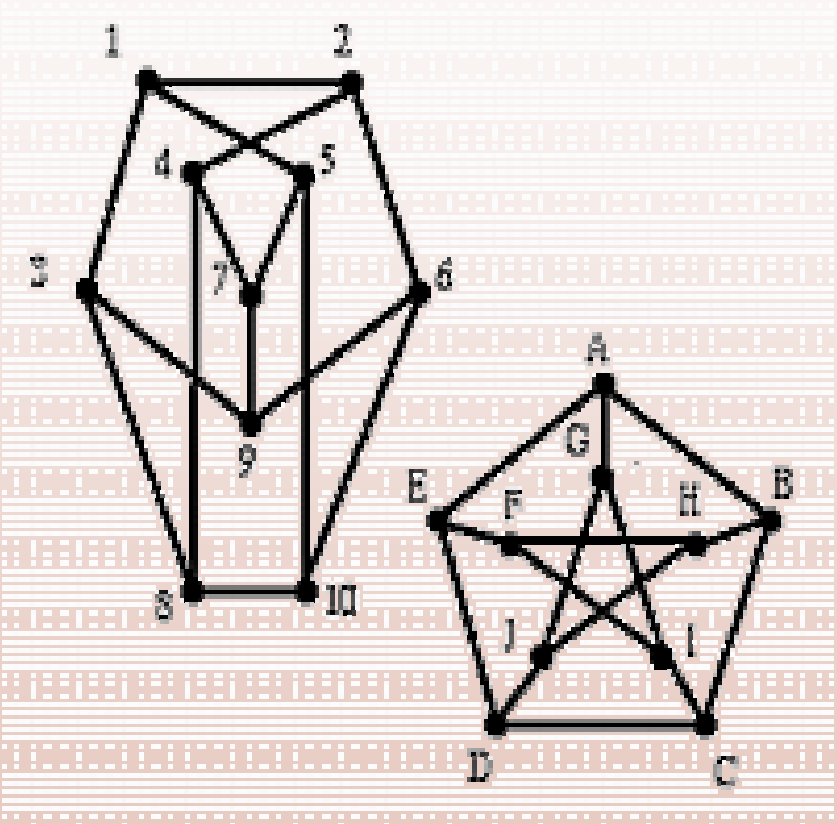
How do these graphs relate?

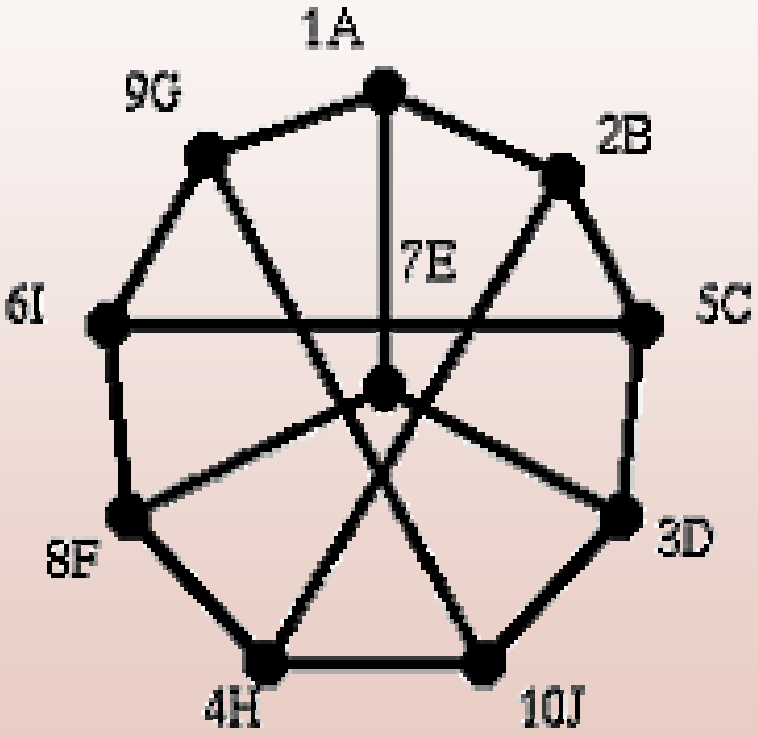


$= \cong \approx \neq$

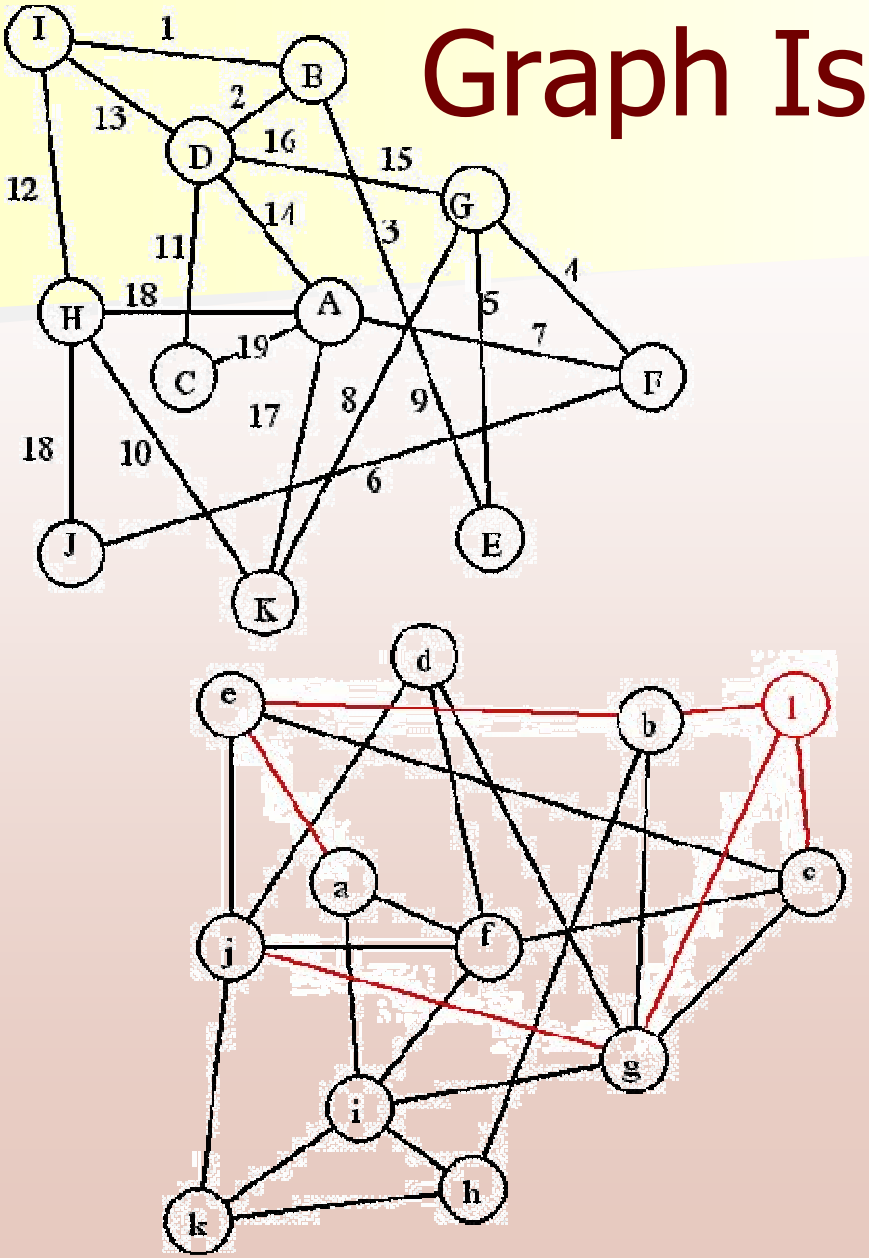


Are these two graphs similar?





Graph Isomorphism



- Let G and G′ be graphs with vertex sets V(G) and V(G′) and edge sets E(G) and E(G′) respectively.
- G is **isomorphic** to G′ if, and only if, there exists a one-to-one correspondences g: $V(G) \rightarrow V(G')$ and $E(G) \rightarrow E(G')$ that preserves edgepoint functions of G and G′

Graph Isomorphism

- To show isomorphic, must show mapping
  - If G and G' have n vertices and m edges
  - The number of one-to-one correspondences
    - From vertices to vertices is n!
    - From edges to edges is m!
  - So total number of pairs is $n! \cdot m!$
  - If $m = n = 20$,
    - There would be $20! \cdot 20! \cong 5.9 \times 10^{20}$ pairs to check
    - Assuming 1 nanosecond per check, 1.9×10^{20} years
- To show not isomorphic show an invariant doesn't hold

Graph Isomorphic Invariants

- Has n vertices
- Has m edges
- Has a vertex of degree k
- Has m vertices of degree k
- Has a circuit of length k
- Has a simple circuit of length k
- Has m simple circuits of length k
- Is connected
- Has an Euler circuit
- Has a Hamiltonian circuit

Graph Isomorphism Examples

Trees

- A graph is **circuit-free** if, and only if, it has no nontrivial circuits.
- A graph is called a **tree** if it is:
  - Circuit-free and
  - Connected
- A **trivial tree** is a graph that consists of a single vertex
- An **empty tree** has no vertices or edges
- A graph is a **forest** if, and only if, it is circuit-free

- Terminal vertex (a leaf) degree 1
- Internal vertex (a branch vertex) has degree >1

Tree Proofs

- For any positive integer n, any tree with n vertices has n – 1 edges
- If G is any connected graph, C is any nontrivial circuit in G, and any one of the edges of C is removed, then the graph remains connected.
- For any positive integer n, if G is a connected graph with n vertices and n – 1 edges, then G is a tree.

Rooted Trees

- One vertex is distinguished from others as **root**
- **Level of vertex** is number of edges along unique path between it and the root
- **Height** of a rooted tree is the maximum level of any vertex in the tree
- **Children** of v are all vertices adjacent to v, but one level farther from the root than v
- Parent / Siblings / Ancestors / Descendants

Binary Tree

- A rooted tree
- Every parent has at most two children
- Each child is designated as either a left child or a right child

- Full binary tree is a binary tree in which each parent has exactly two children
  - If k internal vertices, then 2k+1 total, and k+1 terminal
- Left and right subtrees

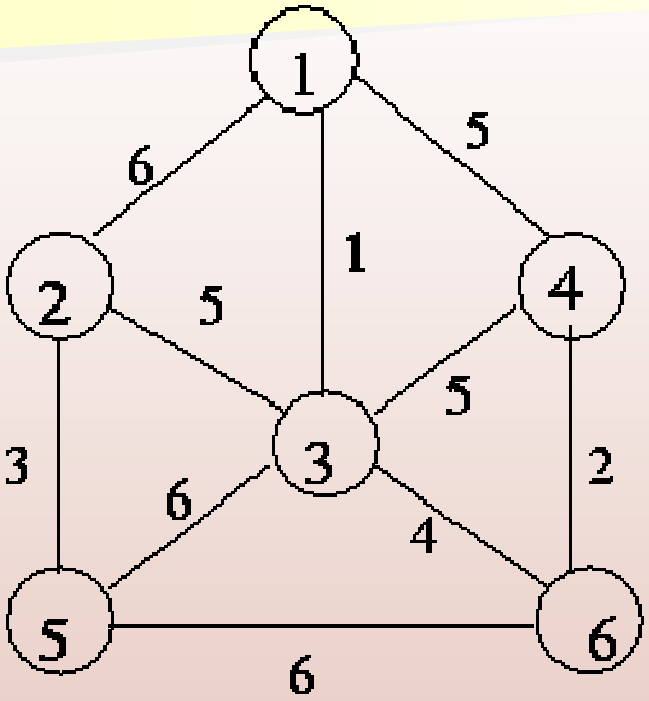
Representing Algebraic Expressions

Spanning Trees

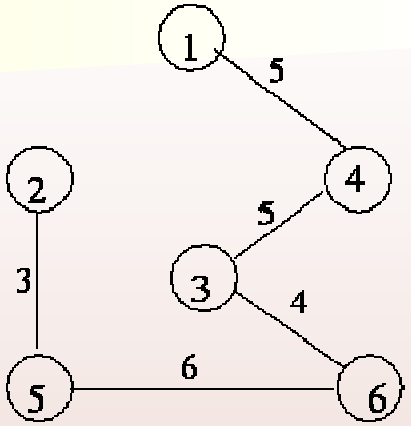
- A spanning tree for a graph G is a subgraph of G that contains every vertex of G and is a tree.

- Every connected graph has a spanning tree.
- Any two spanning trees for a graph have the same number of edges.

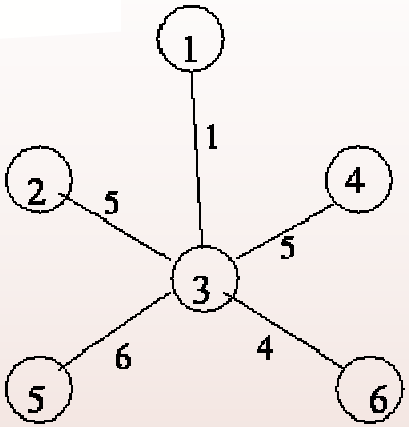
Spanning Trees



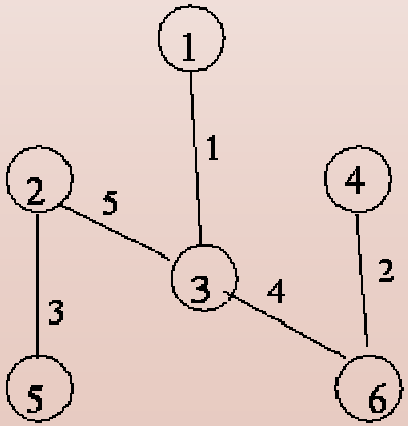
A connected graph.



A spanning tree with cost = 23

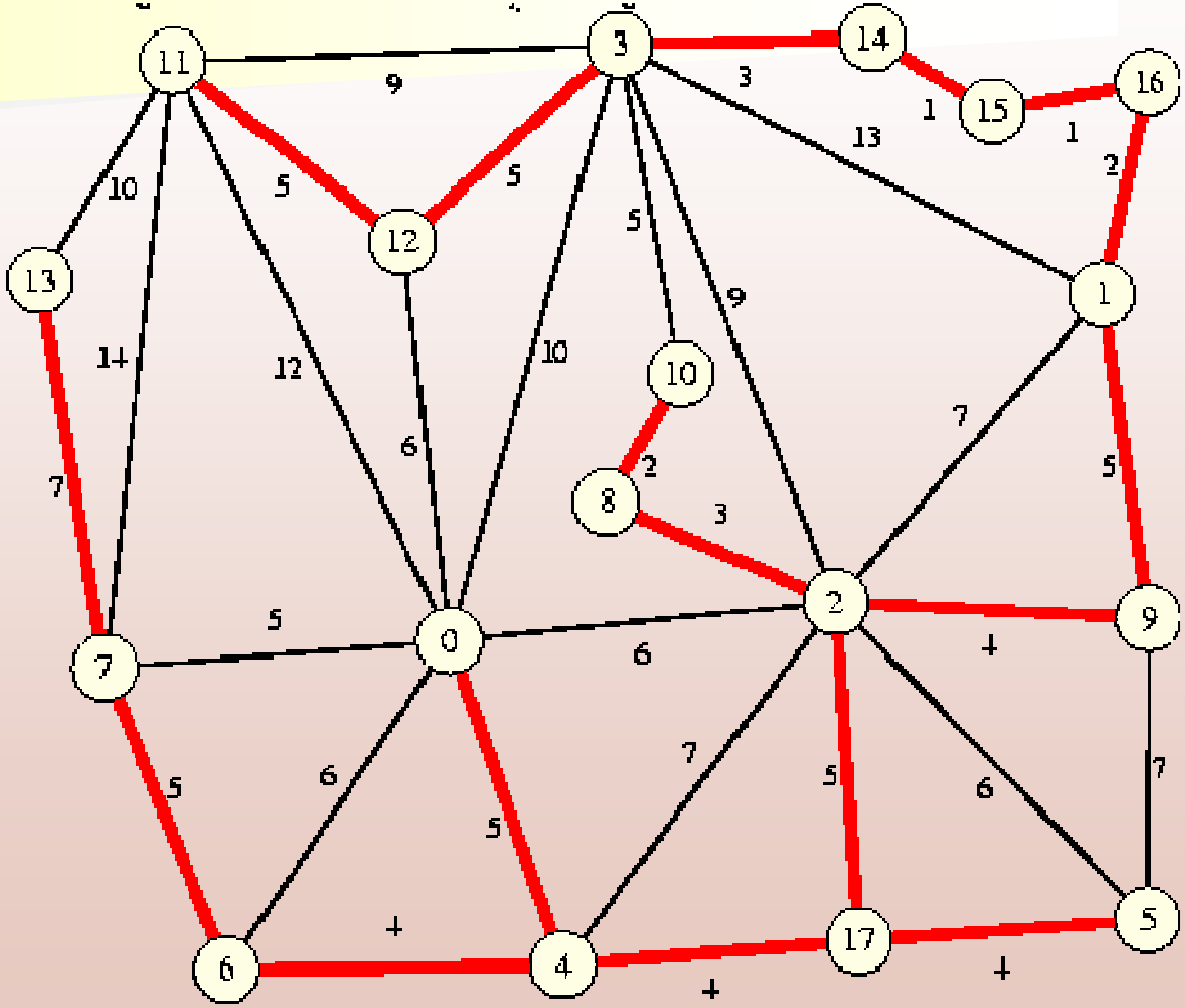


Another spanning tree with cost 21



MST, cost = 15

Minimum Spanning Tree



Kruskal's Algorithm

- The algorithm continuously increases the size of a tree starting with a single vertex until it spans all the vertices.
- Input: A connected weighted graph G(V,E)
- Initialize: V' = $\{v_1, v_2, \ldots, v_n\}$ – all of the vertices of G, E'= {}, n(E') = 0
- While (n(E') < n – 1):
  - Find an edge e in E of least weight
  - Delete e from E
  - If addition of e doesn't produce circuit', add to E'
- Output: G(V',E') is the minimal spanning tree

Prim's Algorithm

- The algorithm continuously increases the size of a tree starting with a single vertex until it spans all the vertices.
- Input: A connected weighted graph G(V,E)
- Initialize: V' = {x}, where x is an arbitrary node from V, E'= {}
- Repeat until V'=V:
  - Choose edge (u,v) from E with minimal weight such that u is in V' and v is not in V' (if there are multiple edges with the same weight, choose arbitrarily)
  - Add v to V', add (u,v) to E'
- Output: G(V',E') is the minimal spanning tree

Proof of Correctness (and Efficiency)

- Correctness
  - See the book
- Worst-case orders of
  - Kruskal's Algorithm – m log m
  - Prim's Algorithm – n^2