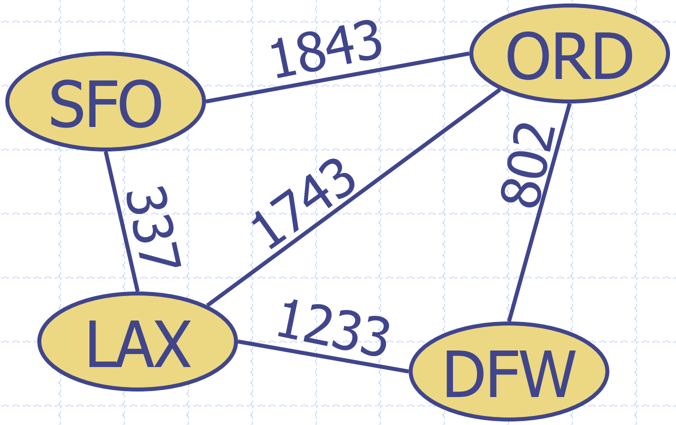
Lesson 11
Graphs:
Combinatorics of Pure Intelligence

Wholeness of the Lesson

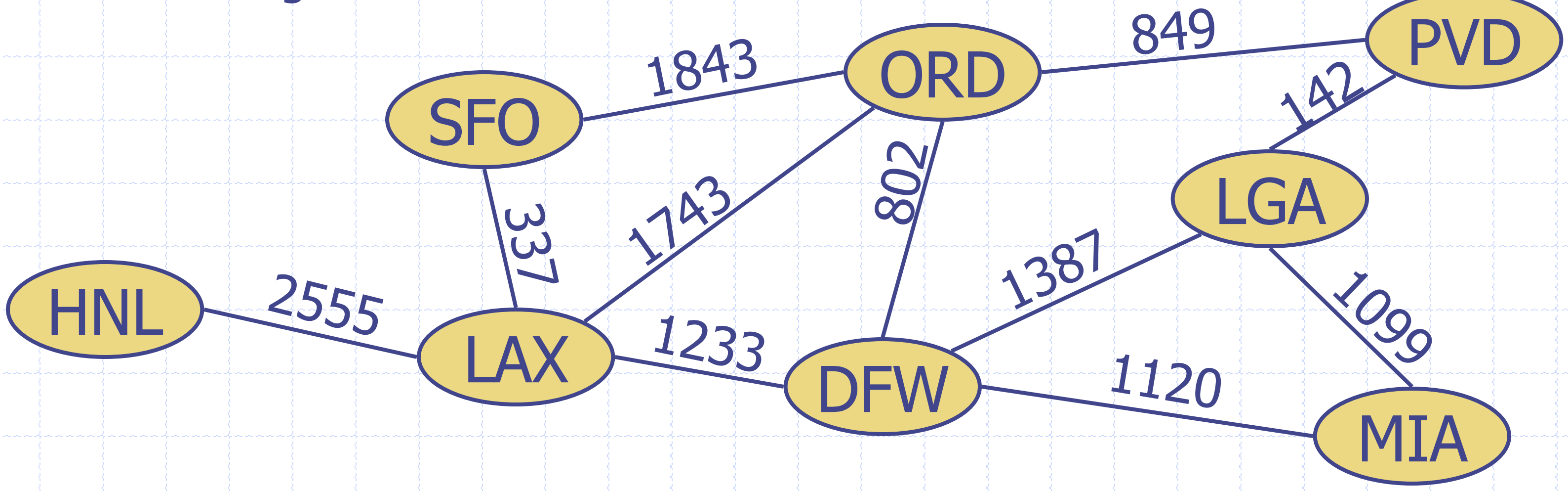
Graphs are data structures that do more than simply store and organize data; they are used to model interactions in the world. This makes it possible to make use of the extensive mathematical knowledge from the theory of graphs to solve problems abstractly, at the level of the model, resulting in a solution to real-world problems.

Science of Consciousness: Our own deeper levels of intelligence exhibit more of the characteristics of Nature's intelligence than our own surface level of thinking. Bringing awareness to these deeper levels, as the mind dives inward, engages Nature's intelligence, Nature's know-how, and this value is brought into daily activity. The benefit is greater ability to solve real-world problems, meet challenges, and find the right path for success.



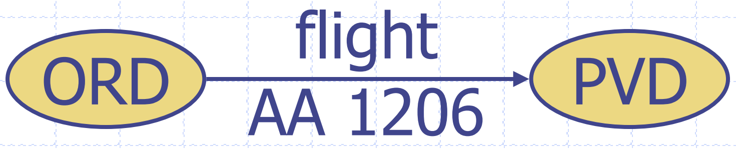
Graph

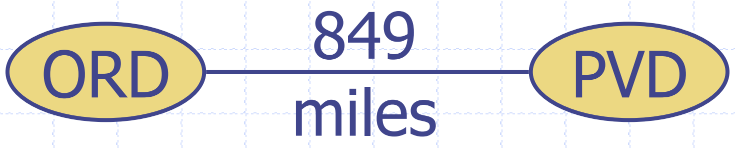
- A graph is a pair (V, E), where
  - V is a set of nodes, called vertices
  - E is a collection of pairs of vertices, called edges
  - Vertices and edges can be implemented so that they store elements
- Example:
  - A vertex represents an airport and stores the three-letter airport code
  - An edge represents a flight route between two airports and stores the mileage of the route



Edge Types

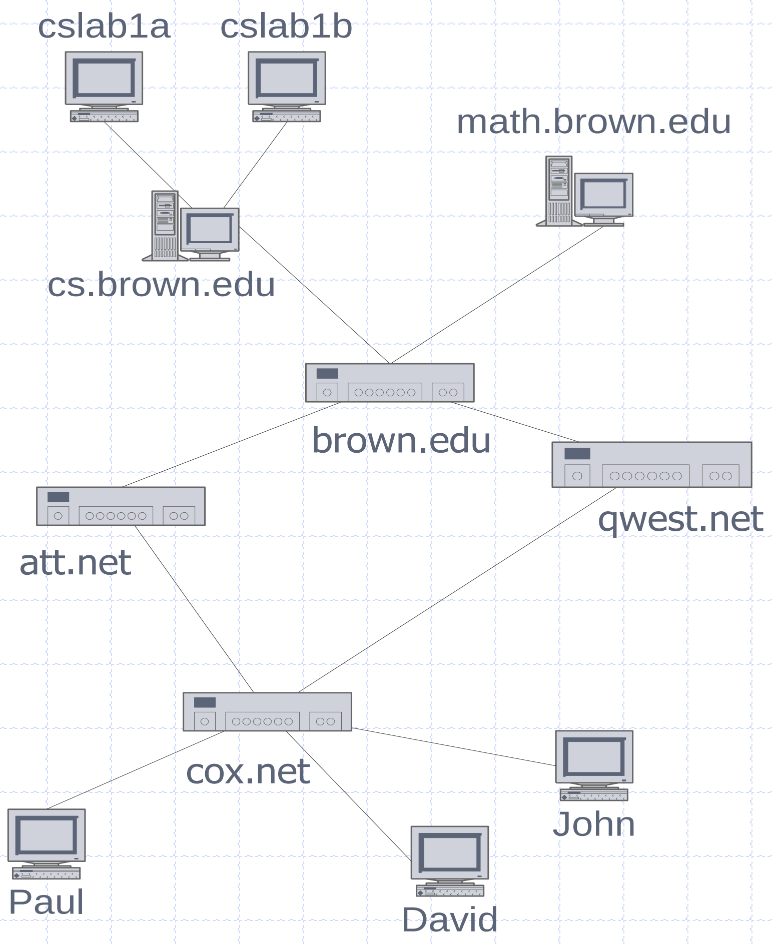
- Directed edge
  - ordered pair of vertices (u,v)
  - first vertex u is the origin
  - second vertex v is the destination
  - e.g., a flight
- Undirected edge
  - unordered pair of vertices (u,v)
  - e.g., a flight route
- Directed graph
  - all the edges are directed
  - e.g., flight network
- Undirected graph
  - all the edges are undirected
  - e.g., route network





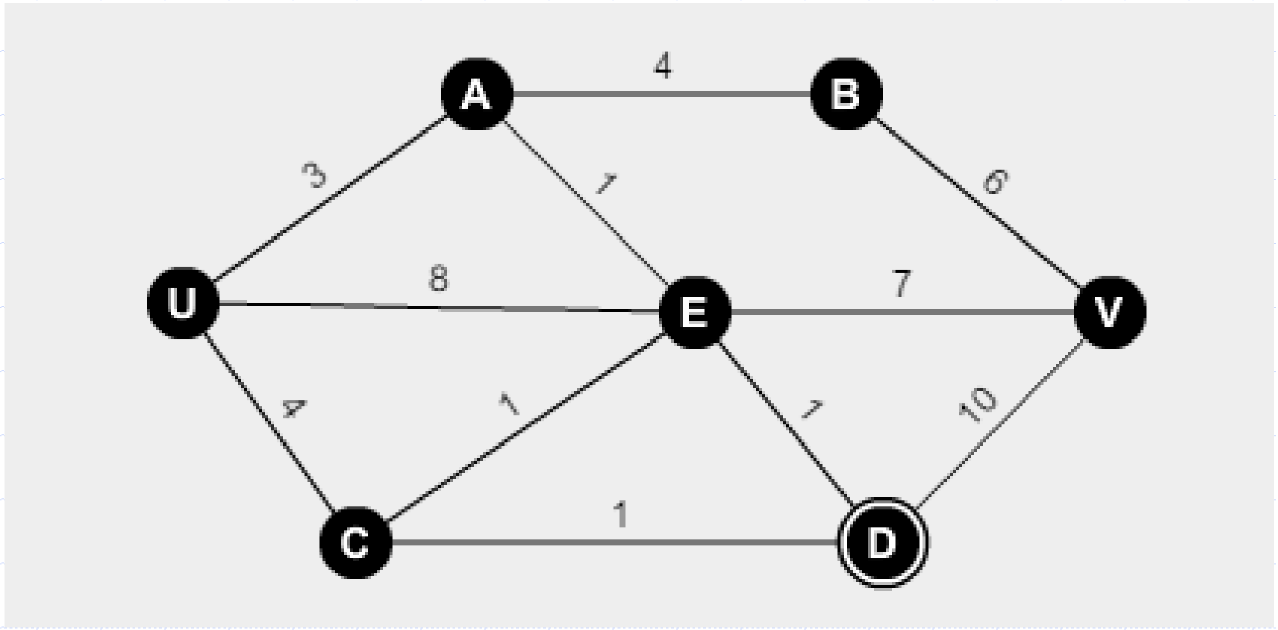
Applications

- Electronic circuits
  - Printed circuit board
    (nodes = junctions, edges are the traces)
- Transportation networks
  - Highway network
  - Flight network
- Computer networks
  - Local area network
  - Internet
  - Web
- Databases
  - Entity-relationship diagram
- Physics / Chemistry
  - Atomic structure simulations (e.g. shortest path algs)
  - Model of molecule -- atoms/bonds



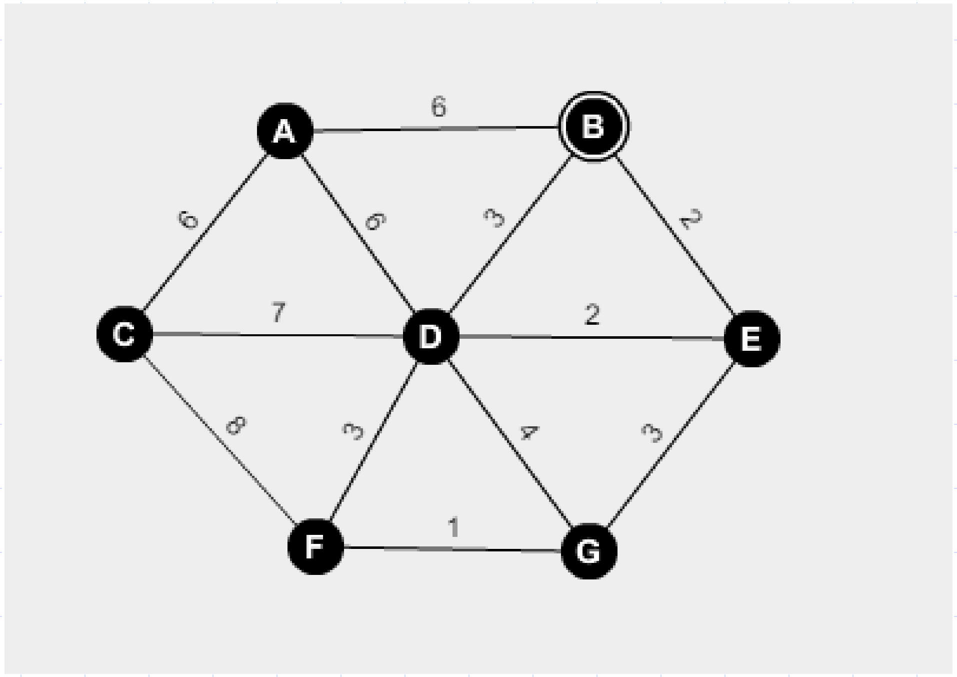
Examples

SHORTEST PATH. The diagram below schematically represents a railway network between cities; each numeric label represents the distance between respective cities. What is the shortest path from city U to city V? Devise an algorithm for solving such a problem in general.



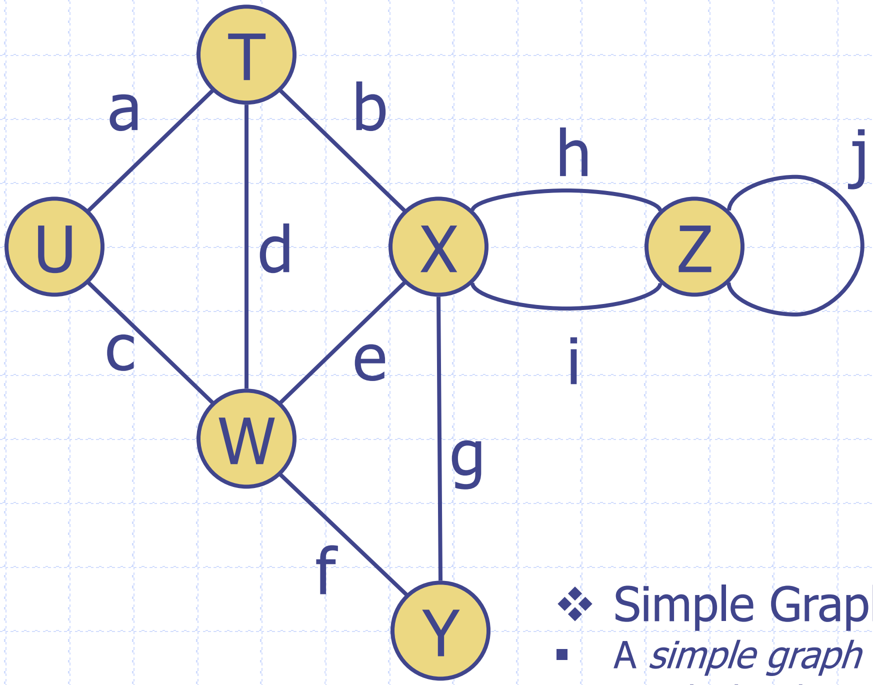
Examples (continued)

CONNECTOR. The diagram below schematically represents potential railway paths between cities; a numeric label represents the cost to lay the track between the respective cities. What is the least costly way to build the railway network in this case, given that it must be possible to reach any city from any other city by rail? Devise an algorithm for solving such a problem in general.



Terminology

- |V| (also v or *n*) is the number of vertices of G; |E| (also ε or *m*) is the number of edges.
- End vertices (or endpoints) of an edge
  - U and T are the endpoints of a
- Edges incident to a vertex
  - a, d, and b are incident to T
- Adjacent vertices
  - U and T are adjacent
- Degree of a vertex
  - X has degree 5 denoted as: deg(X) = 5
- Parallel edges
  - h and i are parallel edges
- Self-loop
  - j is a self-loop



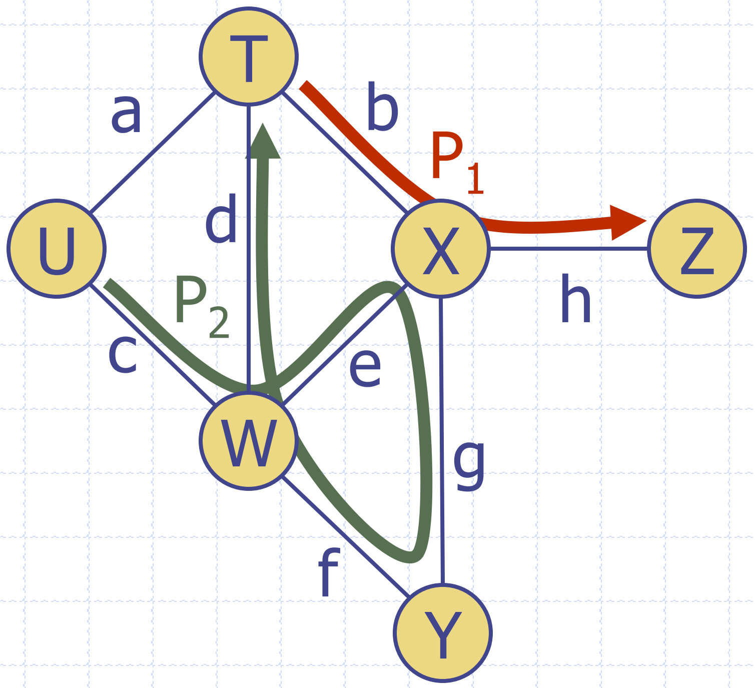
- Simple Graph
  - A *simple graph* is a graph that has no self-loops or parallel edges

Terminology (cont.)

- Path
  - sequence of alternating vertices and edges – in a simple graph, can omit edges

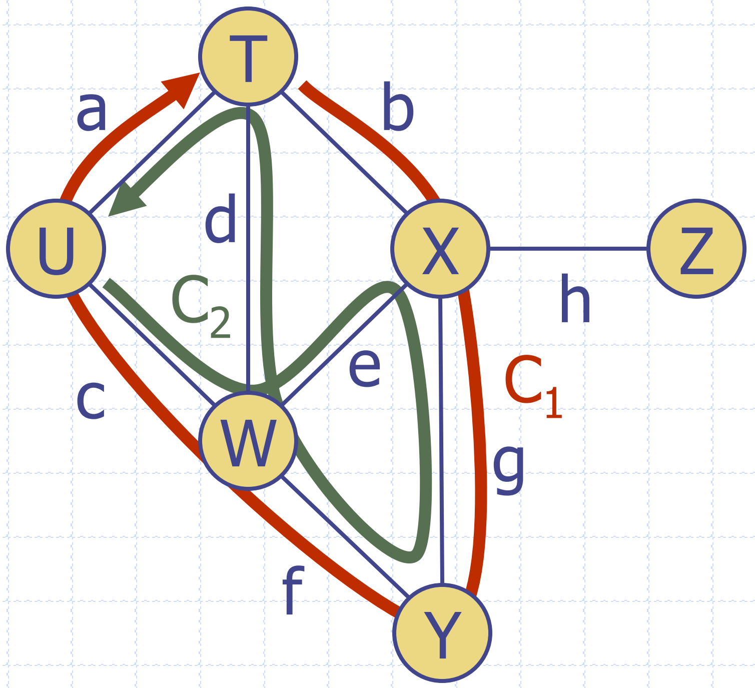
    P_2: U c W e X g Y f W d T

    Or P_2: U W X Y W T
  - begins and ends with a vertex
  - *length* of a path is number of edges
- Simple path
  - path such that all its vertices and edges are distinct
- Examples
  - P_1=(T, X, Z) is a simple path
  - P_2=(U, W, X, Y, W, T) is a path that is not simple



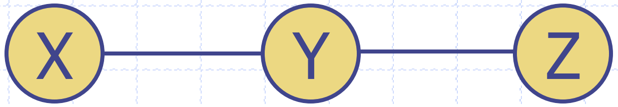
Terminology (cont.)

- Cycle
  - path in which all edges are distinct and whose first and last vertex are the same
  - *length* of a cycle is the number of edges in the cycle
- Simple cycle
  - path such that all vertices and edges are distinct except first and last vertex
- Examples
  - C_1=(T, X, Y, W, U, T) is a simple cycle
  - C_2=(U, W, X, Y, W, T, U) is a cycle that is not simple



Details About Cycles

- Example: Not every path with identical first and last vertex is a cycle (edges must be distinct)



consider X - Y - Z - Y - X

- Facts
  - In a simple graph, the shortest possible cycle has length 3

Properties

Convention: Focus on simple undirected graphs at first

Property 1

$$\Sigma_{\mathbf{v}} \deg(\mathbf{v}) = 2\varepsilon$$

Proof: Let E_v = {e | e incident to v}. Then

$$\Sigma_{\mathbf{v}} \deg(\mathbf{v}) = \Sigma_{\mathbf{v}} |E_{\mathbf{v}}|$$

Notice every edge (v,w) belongs to just two of these sets: E_v and E_w. So every edge is counted exactly twice.

Property 2

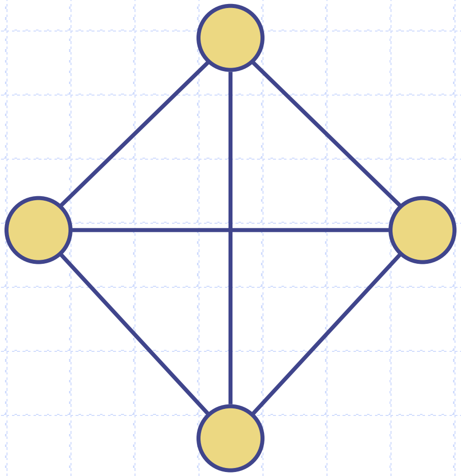
$$\varepsilon \leq \nu(\nu - 1)/2$$

Proof: max number of edges is

$$C(\nu, 2) = C_{\nu,2} = \binom{\nu}{2}$$

Notation

ν	number of vertices
ε	number of edges
$\deg(\mathbf{v})$	degree of vertex $\mathbf{v}$

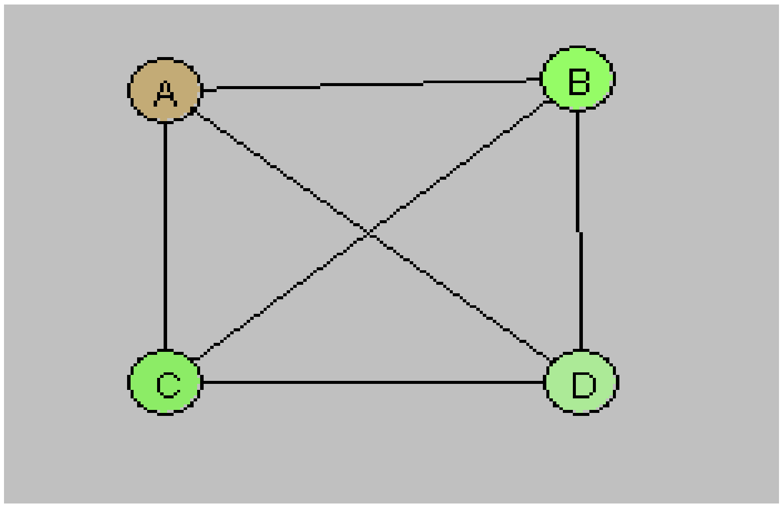


Example

- $\nu = 4$
- $\varepsilon = 6$
- $\deg(\mathbf{v}) = 3$

Complete Graphs

- A graph G is **complete** if for every pair of vertices u, v, there is an edge $e \in E$ that is incident to u, v. In other words, for every u, v, $(u, v) \in E$.
- This is the complete graph on 4 vertices, denoted K_4.



- In general, the complete graph on n vertices is denoted K_n. The one-point graph is K_1.
- **Exercise**. For a complete graph G,

$$\epsilon = \frac{\nu(\nu - 1)}{2}.$$

Therefore, K_n has exactly $n(n-1)/2$ edges.

Subgraphs

- A graph $H = (V_H, E_H)$ is a **subgraph** of a graph $G = (V_G, E_G)$ if both of the following are true:

  a. $V_H \subseteq V_G$ and $E_H \subseteq E_G$, and

  b. for every edge (u, v) belonging to E_H, both u and v belong to V_H.

  H is called a **spanning subgraph** if $V_H = V_G$.

- Suppose $G = (V, E)$ is a graph and $W \subseteq V$, where W is nonempty. Then the subgraph of G that is **induced by** W, denoted $G[W]$, is the subgraph of G whose vertices are W and whose edges are just those edges in E both of whose ends lie in W.

Connected Graphs And Connected Components

- A graph is **connected** if for any two vertices u, v in G, there is a path from u to v.
- **Observation** If a graph $G = (V, E)$ is not connected, then V can be partitioned into sets $V_1, V_2, \ldots, V_k$ (all disjoint) so that each of the subgraphs $G[V_1], G[V_2], \ldots, G[V_k]$ is connected; they are called the **connected components** of G.
- **In-Class Exercise**. Show that if $G = (V, E)$ is a graph, then G is connected whenever
$$\epsilon > \binom{\nu - 1}{2}.$$

Trees

- A graph is **acyclic** if it contains no simple cycle.
- An acyclic connected graph is called a **tree**.
- A **rooted tree** is a tree having a distinguished vertex r.

  The usual levels we have in trees as data structures can be defined for rooted trees. Level 0 contains only the root. Level 1 contains all vertices adjacent to the root. Level 2 contains all vertices adjacent to a Level 1 vertex other than the Level 0 vertex. In general, Level $i+1$ contains vertices adjacent to the Level i vertices that do not appear at previous levels.

  Facts About Levels Suppose T is a rooted tree with levels defined as above.

  A. No two vertices at the same level are adjacent
  B. If v belongs to Level i and w belongs to Level j and $|i-j|>1$ then v and w are not adjacent.
  C. Every vertex but the root has exactly one parent.
  D. If v is in Level i and w is in Level j, $i<j$, and w is a descendant of v, then a path from v to w includes one vertex from every level between Levels i and j.

- A subgraph T of G is a **spanning tree** if T is a spanning subgraph and also a tree.
- **Theorem**. In a tree, any two vertices are connected by a unique simple path.
- **Theorem**. If G is a tree, $\epsilon = \nu - 1$.
- **Exercise**. Suppose G is a graph with $\nu - 1$ edges. Show that the following are equivalent:

    a. G is connected
    b. G is acyclic
    c. G is a tree

Hamiltonian Graphs And Vertex Covers

- A **Hamiltonian cycle** in a graph G is a simple cycle that contains every vertex of G. A graph is a **Hamiltonian graph** if it contains a Hamiltonian cycle.

- If $G = (V, E)$ is a graph, a **vertex cover for** G is a set $C \subseteq V$ such that for every $e \in E$, at least one end of e lies in C.

- **Fact**. The known algorithms for determining whether a graph is Hamiltonian, and for computing the smallest size of a vertex cover, run in exponential time.

Verifying Hard Problems Belong to NP

- Recall from the beginning of the course that we classified the computable functions as being either feasible or infeasible. The feasible ones were those whose running time was bounded by a polynomial in n, and the infeasible ones admitted no such bound.
- In Complexity Theory, *decision problems* rather than *algorithms* are classified in this way. Recall that a decision problem is formulated so that its output is always "true" or "false" (recall the Knapsack problem).
- A decision problem is polynomial-time solvable, and is said to belong to the class $\mathcal{P}$, if there is an alogirthm for solving all instances of the problem of size n in $O(n^k)$ time (for some fixed k).
- Decision problems can be classified in another way (and this classification is useful in cases where no polynomial time solutions are known). A decision problem for which a correct solution can be verified in polynomial time is said to belong to the class $\mathcal{NP}$. Knapsack, VertexCover, HamiltonianCycle can be shown to belong to $\mathcal{NP}$ even though there is no known polynomial-time solution to any of them.

HamiltonianCycle is $\mathcal{NP}$

- We verify that HamiltonianCycle belongs to $\mathcal{NP}$.
- Recall that the input for this problem is a graph $G = (V, E)$, where $|G| = n$. The decision problem to solve is, Does G contain a Hamiltonian cycle?
- To verify that the problem is $\mathcal{NP}$, we assume we have a solution C, which is a subgraph of G. We determine the running time required to verify that C is indeed a Hamiltonian cycle. Here is what must be verified:
  - Check: As a graph, C is a simple cycle.
  - Check: C contains all vertices of G.
  - Check: All edges of C are also edges of G.
- It is not hard to see that these verifications can be performed in $O(n)$ steps. Therefore, HamiltonianCycle belongs to $\mathcal{NP}$.