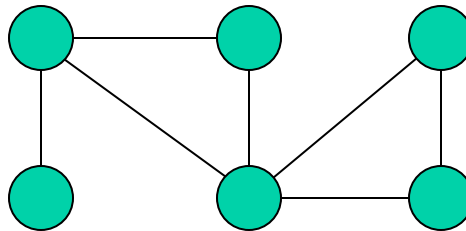


Graphs

A graph G consists of a set of *vertices* V together with a set E of vertex pairs or *edges*.

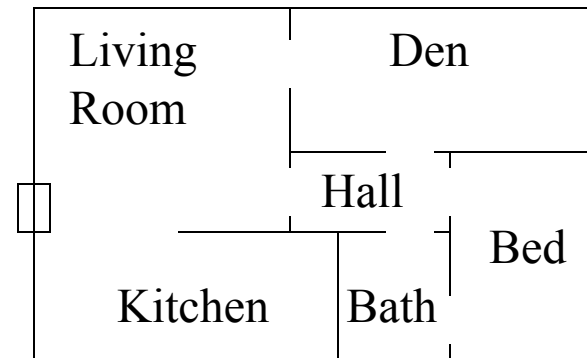
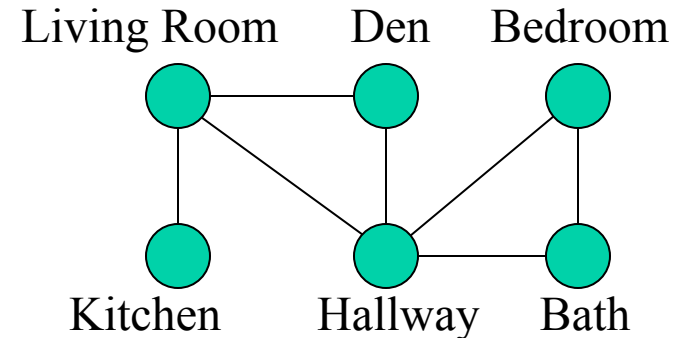
Graphs are important because any binary relation is a graph, so graphs can be used to represent essentially *any* relationship.



What could this mean?

The vertices could represent rooms in a house, and the edges could indicate which of those rooms are connected to each other.

Sometimes a using a graph will be an easy simplification for a problem.

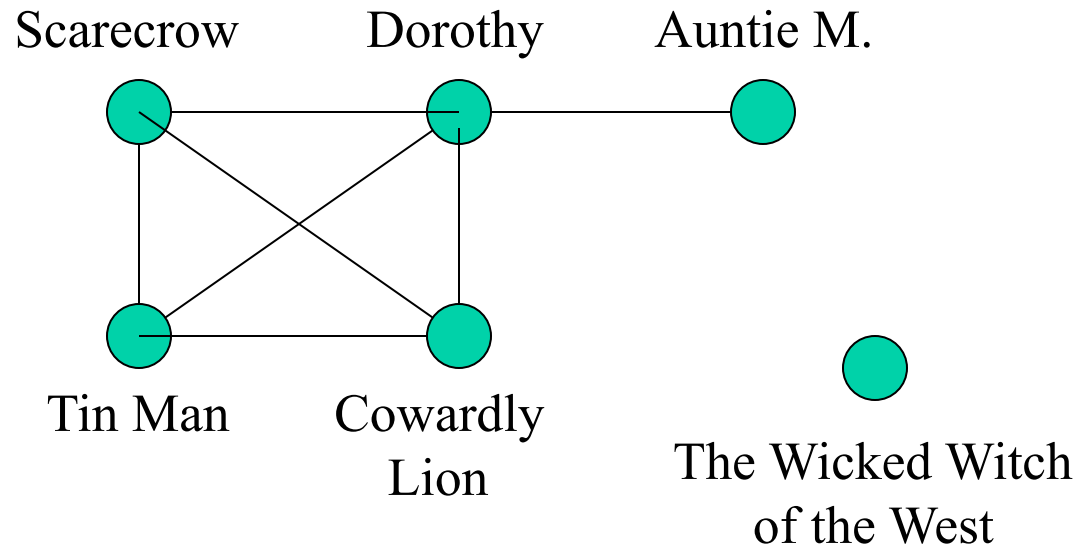


Apartment Blueprint

What else could a Graph mean?

- Vertices are cities and edges are the roads connecting them.
- Edges are the components in a circuit and vertices are junctions where they connect.
- Vertices are software packages and edges indicate those that can interact.
- Edges are phone conversations and vertices are the households being connected.

Friendship Graphs



Each vertex represents a person, and each edge indicates that the two people are friends.

Questions...

If I'm your friend, are you my friend?

A graph is said to be *undirected* if edge (x, y) always implies (y, x) . Otherwise it is said to be *directed*.

Am I my own friend?

An edge of the form (x, x) is said to be a *loop*. If x was y 's friend several times over, we can model this relationship using *multiedges*. A graph is said to be *simple* if it contains no loops or multiedges.

How close a friend are you?

A graph is said to be *weighted* if each edge has an associated numerical attribute. In an *unweighted* graph, all edges are assumed to be of equal weight.

More questions...

Am I linked by some chain of friends to someone famous?

A *path* is a any sequence of edges that connect two vertices. A *simple path* never goes through any vertex more than once. The *shortest path* is the minimum number edges needed to connect two vertices.

Is there a path connecting every two people in the world?

The “six degrees of separation” theory argues that there is always a short path between any two people in the world. A graph is *connected* if there is there is a path between any two vertices. A directed graph is *strongly connected* if there is always a directed path between vertices. Any subgraph that is connected can be referred to as a *connected component*.

Still More Questions...

Who has the most and who has the fewest friends?

The *degree* of a vertex is the number of edges connected to it. The most popular person will have a vertex of the highest degree. Remote hermits may have degree-zero vertices. In *dense* graphs, most vertices have high degree. In *sparse* graphs, most vertices have low degree. In a *regular graph*, all vertices have exactly the same degree.

What is the largest clique?

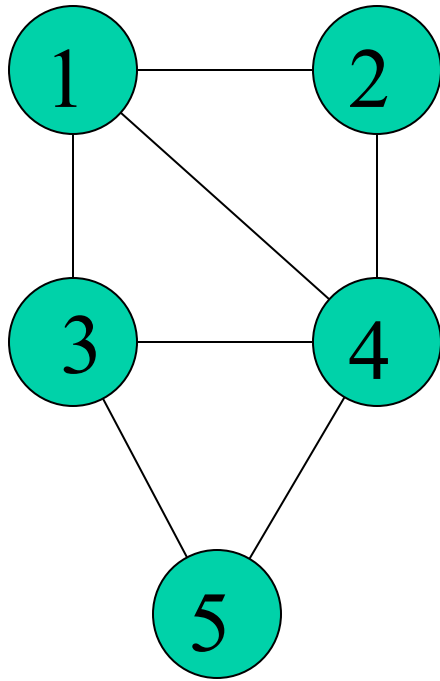
A graph is called *complete* if every pair of vertices is connected by an edge. A *clique* is a sub-graph that is complete.

Yet Another Question...

How long will it take for my gossip to get back to me?

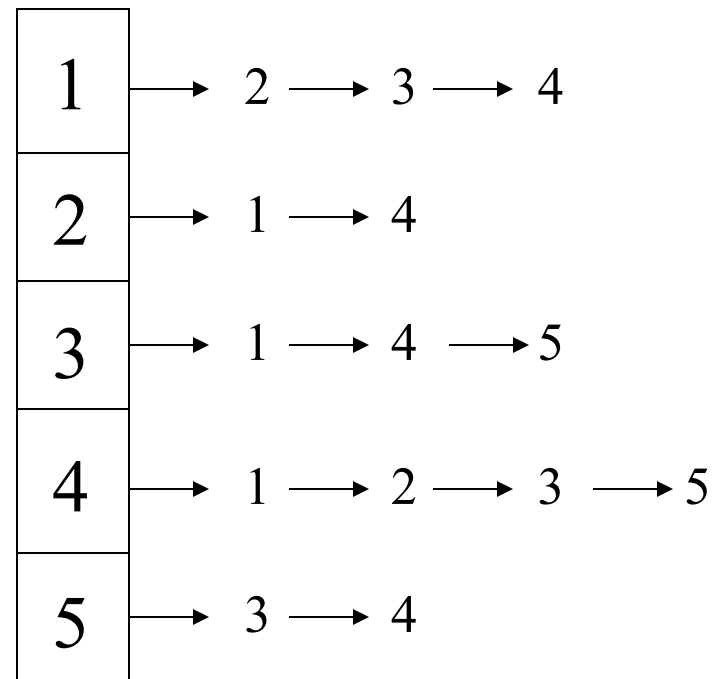
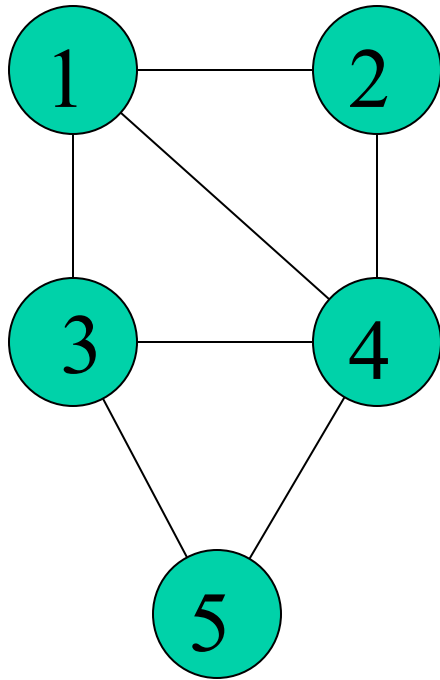
A *cycle* is a path where the last vertex is adjacent to the first. A cycle in which no vertex is repeated is said to be a *simple cycle*. The shortest cycle in a graph determines the graph's *girth*. A simple cycle that passes through every vertex is said to be a *Hamiltonian cycle*. An undirected graph with no cycles is a *tree* if it is connected, or a *forest* if it is not. A directed graph with no directed cycles is said to be a *directed acyclic graph* (or a DAG)

Adjacency Matrices



	1	2	3	4	5
1	0	1	1	1	0
2	1	0	0	1	0
3	1	0	0	1	1
4	1	1	1	0	1
5	0	0	1	1	0

Adjacency Lists



Tradeoffs Between Adjacency Lists and Adjacency Matrices

Comparison

Faster to test if (x, y) exists?

matrices: $\Theta(1)$ vs. $\Theta(m)$

Faster to find vertex degree?

lists: $\Theta(1)$ vs. $\Theta(n)$

Less memory on sparse graphs?

lists: $\Theta(m+n)$ vs. $\Theta(n^2)$

Less memory on dense graphs?

matrices: (small win)

Edge insertion or deletion?

matrices: $\Theta(1)$ vs. $\Theta(m)$

Faster to traverse the graph?

lists: $\Theta(m+n)$ vs. $\Theta(n^2)$

Better for most problems?

lists

Problem:

The square of a directed graph $G = (V, E)$ is the graph $G^2 = (V, E^2)$, such that $(x, y) \in E^2$ iff, for some z , both (x, z) and $(z, y) \in E$; i.e., there is a path of exactly two edges.

Give efficient algorithms to square a graph on both adjacency lists and matrices.

G^2 with Adjacency Matrices

Given an adjacency matrix, we can check in constant time whether a given edge exists. To discover whether there is an edge (x, y) in E^2 , for each possible intermediate vertex z we can check whether (x, z) and (z, y) exist in $O(1)$.

Since there are $O(n)$ intermediate vertices to check, and $O(n^2)$ pairs of vertices to ask about, this takes $O(n^3)$ time.

G^2 with Adjacency Lists

For a given edge (x, z) , we can run through all the edges from z in $O(n)$ time, and fill the results into an adjacency matrix of G^2 , which is initially empty.

It takes $O(mn)$ to construct the edges, and $O(n^2)$ to initialize and read the adjacency matrix, for a total of $O((n + m)n)$. Since $m+1 \geq n$ (unless the graph is disconnected), this is usually simplified to $O(mn)$, and is faster than the previous algorithm on sparse graphs.