Graph $G$: Formal Definition

- A graph $G$ consists of two sets $G = \{V, E\}$
  - A set $V$ of vertices, or nodes (entities)
  - A set $E \subseteq V \times V$ of edges (relationships between entities)

Examples of Graphs

Graphs representing relationships between individuals and proteins.
Graph G: Formal Definition

- A subgraph
  - Consists of a subset of a graph’s vertices and a subset of its edges
- Adjacent vertices
  - Two vertices that are joined by an edge

Paths and Cycles

- A path between two vertices
  - A sequence of edges that begins at the first vertex and ends at the other vertex
- A simple path
  - A path that passes through a vertex at most once
- A cycle
  - A path that begins and ends at the same vertex
- A simple cycle
  - A cycle that does not pass through a vertex more than once

Complete Graph

- A complete graph
  - A graph that has an edge between each pair of distinct vertices
  - How many edges does a complete graph with n nodes have?

Connected and Disconnected

- A connected graph
  - A graph that has a path between each pair of distinct vertices
- A disconnected graph
  - A graph that has at least one pair of vertices without a path between them
Directed Graphs and DAGs

- **Directed graph**
  - Each edge is a directed edge, or an arc, or a link
  - Can have two arcs between a pair of vertices, one in each direction
  - Vertex \( y \) is adjacent to vertex \( x \) if and only if there is a directed edge from \( x \) to \( y \)

- **Directed path**
  - A sequence of directed edges between two vertices

- **Directed Acyclic Graph (DAG)**
  - Directed graph that has no cycles

How few arcs can you remove to make the graph a DAG?

Weighted Edges

- **Weighted graph**
  - A graph whose edges have weights
  - Usually weight correspond to the "cost" of the relationship represented by the edge

Multiple Edges

- **Multigraph**
  - Not a graph
  - Allows multiple edges between vertices
  - Multiple edges indicate multiple relations between vertices

Tree: A Special Graph

A **tree** is a connected graph in which there is exactly one simple path connecting any two nodes

How many edges does a tree with \( n \) nodes have?
public interface Graph<T> { // partial

public boolean isEmpty() // returns true iff a graph is empty
public int n() // returns the number of vertices in a graph
public void addVertex(T v) // Insert a vertex in a graph
public void removeVertex(T v) // Deletes a vertex from a graph along with any edges between the vertex and other vertices
public void addEdge(T v1, T v2) // Insert an edge between two given vertices in a graph
public void removeEdge(T v1, T v2) // Deletes the edge between two given vertices in a graph
public T findVertex(String key) // Retrieves and returns the vertex that contains a given search key
public boolean isEdge(T v1, T v2) // returns true iff an edge exists between two given vertices
public LinkedList<T> getNeighbors(T v) // FOR UNDIRECTED GRAPH
  // Retrieves and returns a list of the vertices adjacent to vertex v

Implementing Graphs: Adjacency Matrix

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Implementing Graphs: Adjacency Matrix

Visualizing Graphs with yEd

- yEd: A great and simple graph visualization
- You can create any graph by clicking (for vertices) and clicking-and-dragging (for edges)
- Lots of graph formats supported. Use .tgf
- TGF format: a text file listing lines of:
  - vertexID: vertexName (for vertices)
  - #
  - vertexID pairs (for arcs)
- Once you upload a file, choose Layout > Circular to see it laid out nicely.

What property does the matrix of an undirected graph have?
import java.util.*;
import java.io.*;

public class AdjMatGraph<T> implements Graph<T>
{
    private final int NOT_FOUND = -1;
    private final int DEFAULT_CAPACITY = 1; // 1, to test expandCapac.

    private int n; // number of vertices in the graph
    private boolean[][] arcs; // adjacency matrix of arcs
    private T[][] vertices; // values of vertices

    (cont.)

How would you implement these?

public boolean isEmpty() // returns true iff a graph is empty
public int n() // returns the number of vertices in a graph
public int m() // returns the number of edges in a graph

Implementing Graphs: Adjacency Lists

- An adjacency list for a graph with n vertices numbered 0, 1, ..., n – 1
  - Consists of n linked lists
  - The i-th linked list has a list entry for vertex j
    iff the graph contains an arc from vertex i to vertex j

Adjacency Matrix vs Adjacency Lists

- Which representation supports better these two frequent operations on graphs?
  - isEdge(v, w)
    Determine whether there is an edge from vertex v to vertex w
  - getNeighbors(v)
    Return list of all vertices linked to from a given vertex v
    - Adjacency matrix
      - Supports isEdge(v, w) more efficiently
    - Adjacency list
      - Supports getNeighbors(v) more efficiently
      - Often requires less space than an adjacency matrix
Weighted Adjacency Matrix

- Adjacency matrix for a weighted graph with $n$ vertices numbered 0, 1, ..., $n-1$
  - An $n \times n$ array matrix $EdgeW$ such that $EdgeW[i][j]$:
    - The weight of the arc from vertex $i$ to vertex $j$
    - $\infty$ if there is no edge from vertex $i$ to vertex $j$

```
Weighted Adjacency Lists

- Adjacency list for a weighted undirected graph
  - Each list entry contains the edge label and weight
  - Treats each edge as if it were two arcs in opposite directions
```