Graph definitions and terminology

A graph contains nodes (also called vertices) and edges. An edge connects two nodes.

We can use graphs to represent relationships: there is a relationship between node A and node B if there is an edge between A and B.

In an undirected graph, relationships are mutual.

In a directed graph, relationships are one-way, from the source to the destination.

In a weighted graph, relationships have associated information (e.g., cost or “weight”).

An edge is adjacent to A if that edge emanates from A.

Node B is adjacent to node A if there is an edge from A to B.

The neighbors of A are all the nodes adjacent to A.

A path is a sequence of edges between adjacent nodes.

Node B is reachable from node A if there is a path from A to B.

In a complete graph, there exists an edge between each pair of nodes.

In a connected graph, there exists a path between each pair of nodes.

A sparse graph has few edges, relative to the maximum amount it can have.

A dense graph has many edges, relative to the maximum amount it can have.

In an acyclic graph, there are a finite number of paths between any two nodes.

In a cyclic graph, there may be an infinite number of paths between two nodes.

Designing and implementing a new data structure

The interface describes what a data structure can do (e.g., its operations). The interface is a promise from the provider of the data structure to the user of the data structure.

The implementation describes how the data structure works (e.g., how the data are stored / organized and which algorithms are used to provide the operations). The implementation makes good on the promise of the interface.

It should be possible to replace the implementation without modifying the interface.

Next time: Representing graphs and implementing them in Java