Trees

A graph is a data structure made from vertices and edges. An edge notionally connects two vertices.

\[ \text{VERTEX} \]
\[ \text{EDGE} = \text{VERTEX} \times \text{VERTEX} \]

A tree is a special case of a graph: it has exactly \( n \) vertices and \( n - 1 \) edges, and is connected.

As a consequence of this definition, there can be no cycles in a tree.

We can think of trees as directed or undirected, but normally undirected. In a directed tree, the edges are read left-to-right: in an undirected tree, they are considered symmetric, or are considered to run in both directions.

A rooted tree has a distinguished root vertex. We refer to a vertex as the parent of a neighbor (child) if it is closer to the root of the tree.
Binary Trees

A binary tree is a rooted tree such that the root vertex has exactly two neighbors, and all other vertices have three. We usually draw these “upside-down” for convenience.

Graphs

A graph is the general case where there can be cycles.

\[
\begin{array}{c}
\text{Graph} \\
\text{vertices} : \mathbb{P} \text{ VERTEX} \\
\text{edges} : \mathbb{P} \text{ EDGE}
\end{array}
\]

Again connected, directed, undirected, rooted etc are defined in the obvious way.

Labeling

The edges and/or vertices of a graph may have labels attached. The labels can be thought of as given by a labeling function, in the fashion we are used to.

Graph Representation

There are actually multiple standard ways to represent a graph for modeling or computation:

- Edge List
- Adjacency List
- Adjacency Matrix

Graph Algorithms

Some standard algorithms are useful for manipulating graphs.

- Depth-First Search
- Breadth-First Search
- Transitive Closure