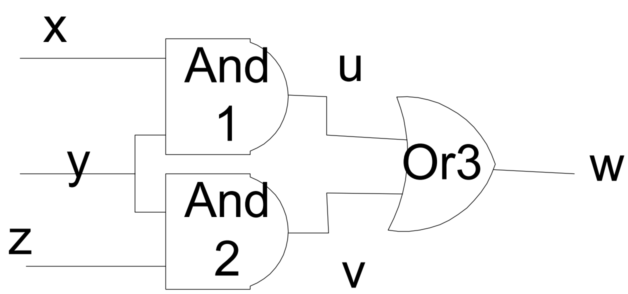
Applications of graphs

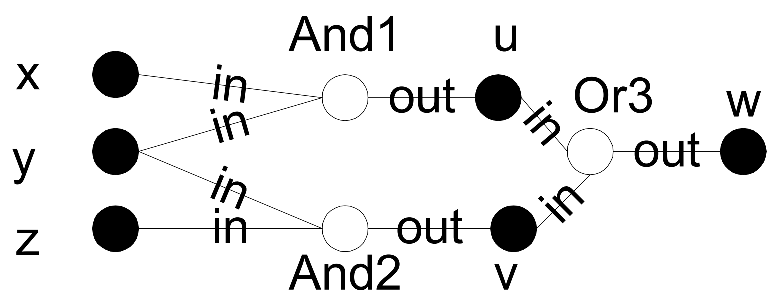
Topological Structure of Circuit

A circuit consists of components and nets.



We can model components and nets as vertices.

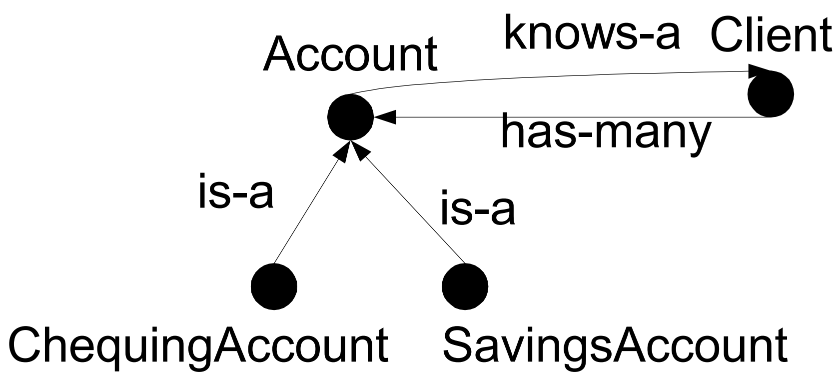
Edges are labeled with names of "ports"



Topological structure of software

Components are classes.

Edges represent relationships between classes.



Structured text

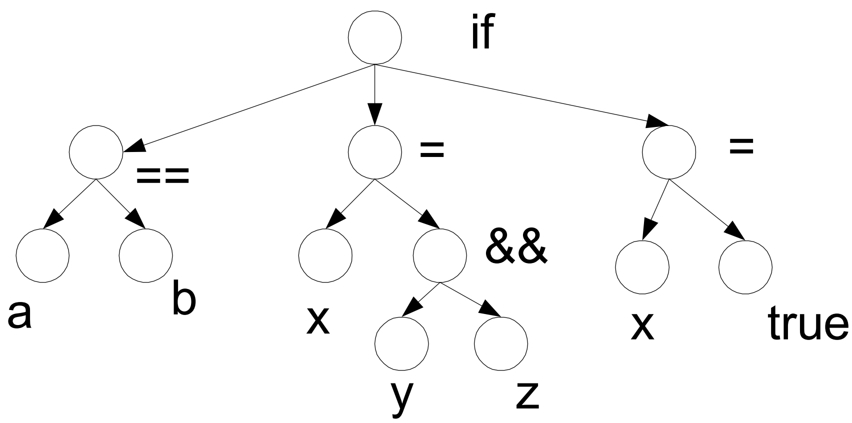
A graph that is connected and has no cycles is called a tree.

Text documents often encode an underlying tree structure.

Examples include HTML, XML, and programming languages.

Text: if(a==b) x = y&&z ; else x = true ;

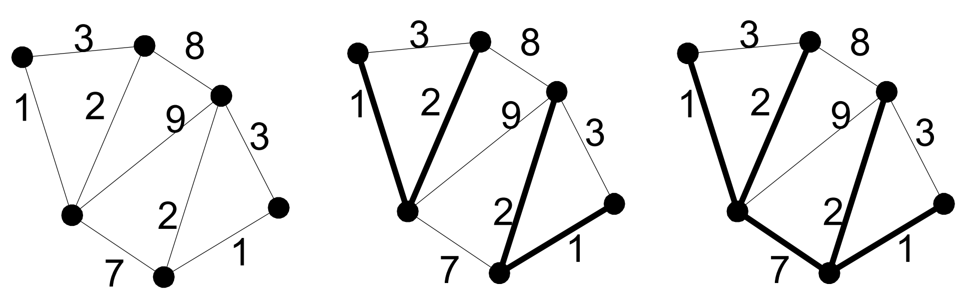
Tree:



Weighted graphs and minimal spanning trees

Suppose you need to put power lines (or fibre optic cable) on a island. The source and each house needs to be connected. There are a number of possible routes based on existing infrastructure, each with an associated positive cost. We need the minimal-weight connected sub-graph that includes every vertex.

An undirected graph that is connected and has no cycles is called an **undirected tree**. Thus we need a *minimal spanning tree*

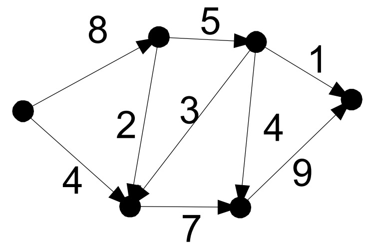


Algorithm (Kruskal): Start with T an empty graph. Consider the edges in order of increasing weight. Add an edge to T if it does not create a cycle

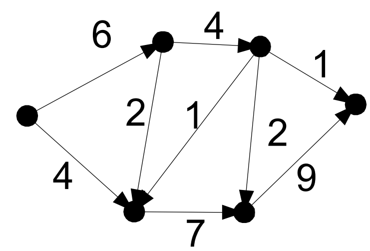
Weighted graphs and max flow

Consider a power network consisting of 1 generating station (**source**), 1 destination station (**sink**), any number of intermediate stations, distribution lines each labelled with a capacity.

We model with a weighted directed graph:



Capacities



A maximal flow

A **flow** is a labelling of the edges such that.

- No edge exceeds its capacity
- Each nonsink/nonsource vertex has equal in-flow and out-flow.

The *maximum flow problem* is to find a flow that maximized the flow into the sink.

A **cut** is a set of edges, the removal of which, disconnects the source from the sink.

Theorem (Ford & Fulkerson): The value of the maximal flow equals the minimum capacity of any cut.