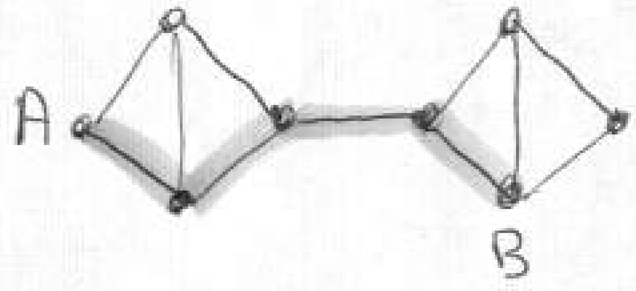
Ch. 7: The Mathematics of Networks

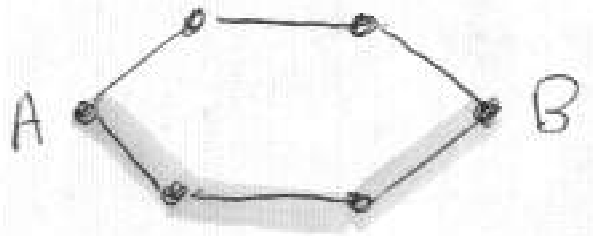
A network is another name for a connected graph.

The degree of separation of two vertices in a network is the length of a shortest path joining the two vertices.

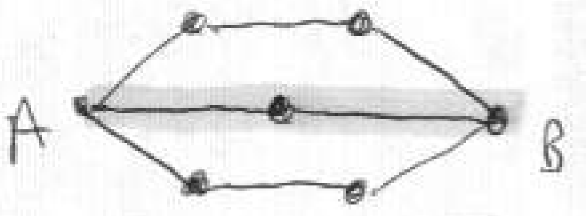
Examples:



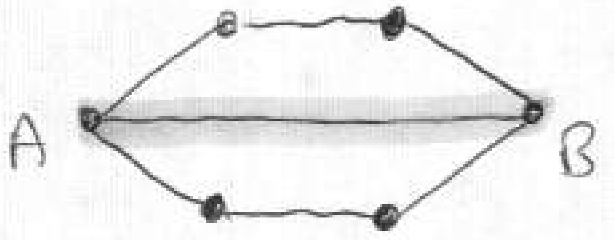
The degree of separation between A and B is 4.



Here it is 3.



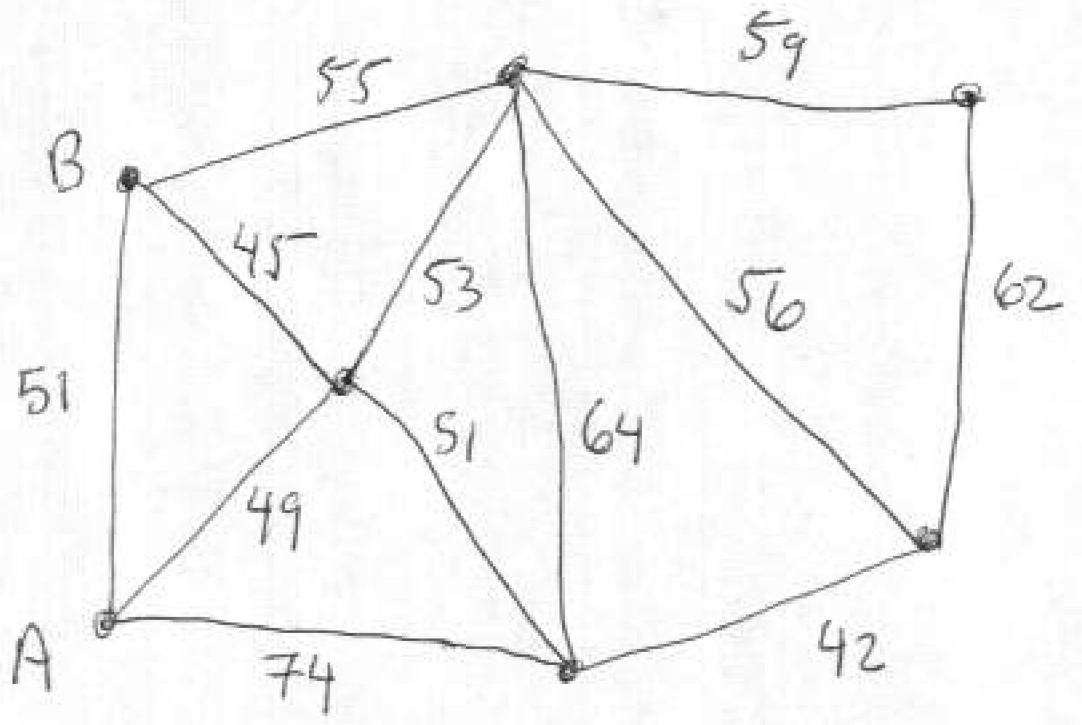
Here it is 2.



Here it is 1.

A typical problem we'll consider in this chapter is the following.

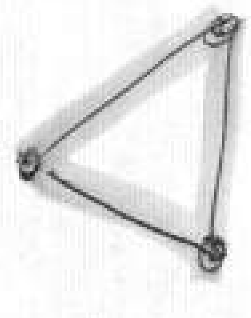
An internet service provider is laying cable in a developing country. Below is a graph (network) in which each vertex represents a location to which the company wants to provide service, and each edge represents a possible route to lay cable.



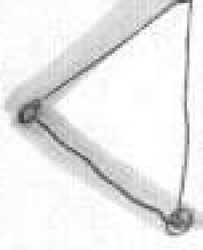
(This example is in the text.) To each edge we have written the cost of laying cable along that route, based on distance and terrain.

The company wants to find the cheapest solution; this means choosing a subgraph (collection of edges) so that every vertex is connected to every other, and with the lowest cost possible

Unlike the previous chapters, here we will want to avoid circuits. This is because circuits are not optimal in this kind of problem. For example, if 3 locations we to be connected



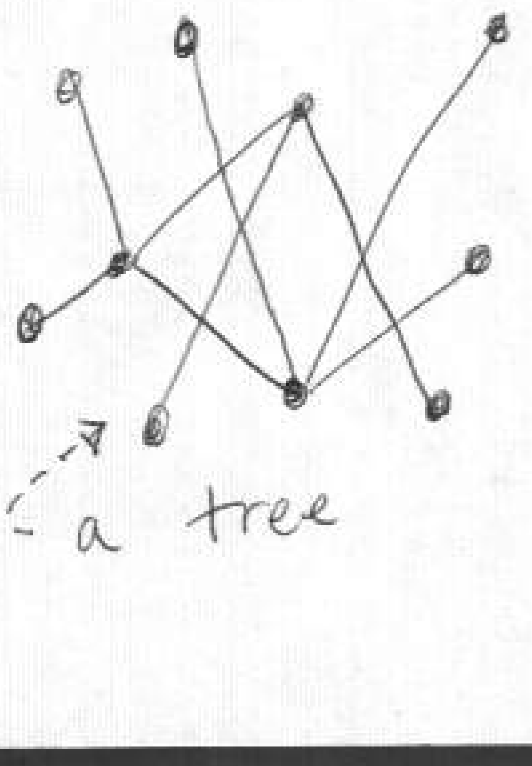
← this solution is more expensive than →



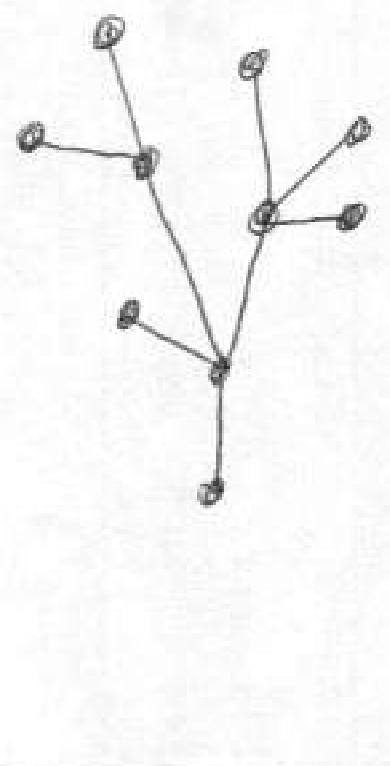
or any other solution with only two edges.
So when creating an optimal network we are creating redundancies and overpaying when circuits are present.

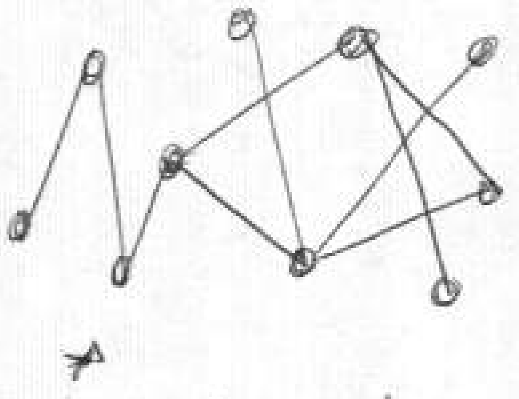
Defn. A tree is a network that has no circuits.

Examples:



We can redraw this to make it look more like a "tree":

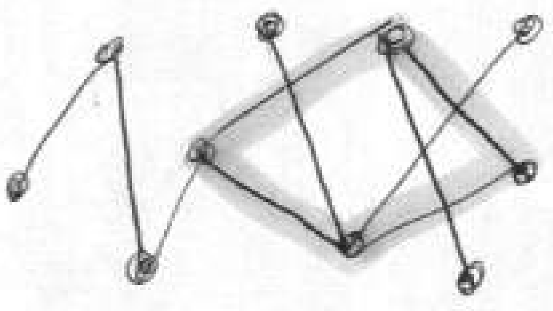




not a tree

Can you find a circuit?

here's one:



Trees have several nice properties.

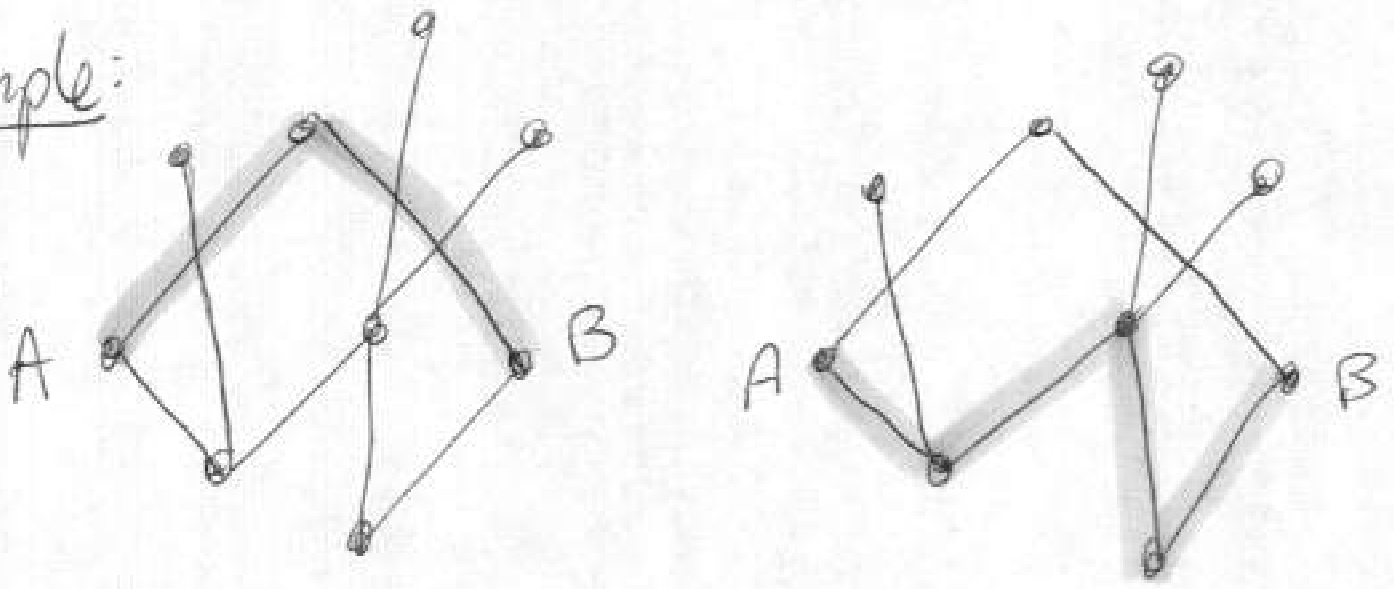
(1) The Single-path property: in a tree, between any two vertices there is exactly one single path connecting them (up to direction).

(2) The All-Bridges property: in a tree, every edge is a bridge.

(3) The N-1 edges property: in a tree with N vertices there are exactly N-1 edges in total.

Property (1) is pretty easy to explain: if there were two distinct paths between two vertices, then they would combine to make a circuit, but a tree can have no circuits.

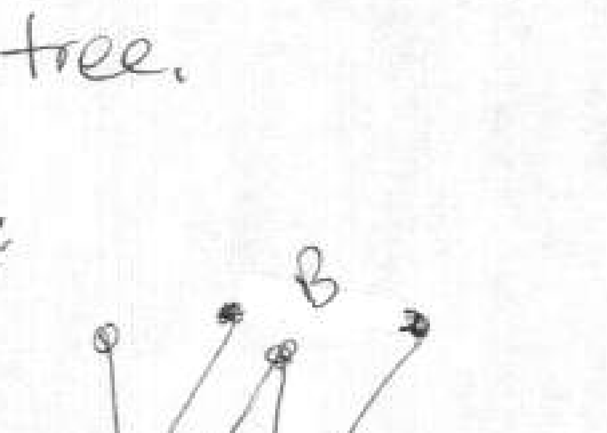
Example:



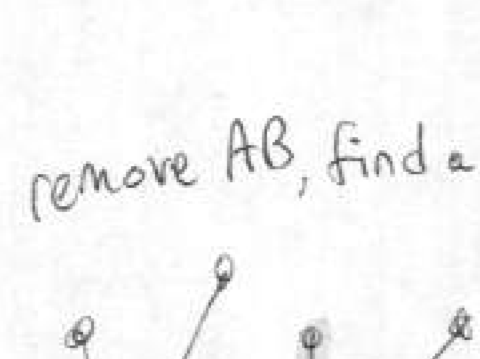
There are 2 distinct paths connecting A to B. If we put them together, we get a circuit, implying that the graph is not a tree.

Property (2) is also rather simple to explain: Suppose there is an edge that is not a bridge. Then upon removing that edge, the graph is still connected. So we can find a path between the two vertices of the edge in the graph that doesn't contain the edge. This path, combined with the edge itself, yields a circuit, showing that our graph is not a tree.

Example:



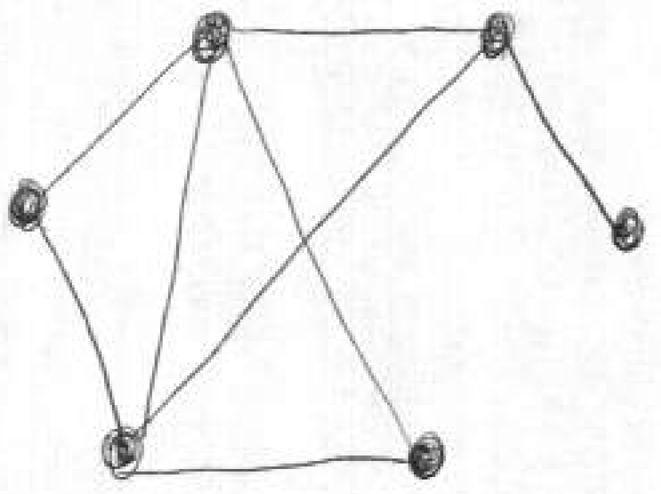
remove AB, find a path, A to B;



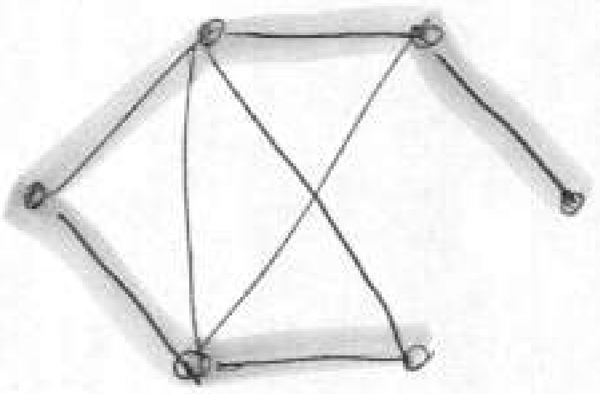
If we put this back in the origi
and add edge A
we get a circui

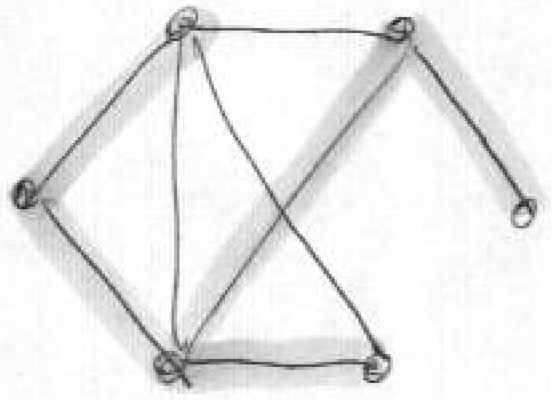
Defn. A spanning tree in a network is a subtree that touches every vertex.

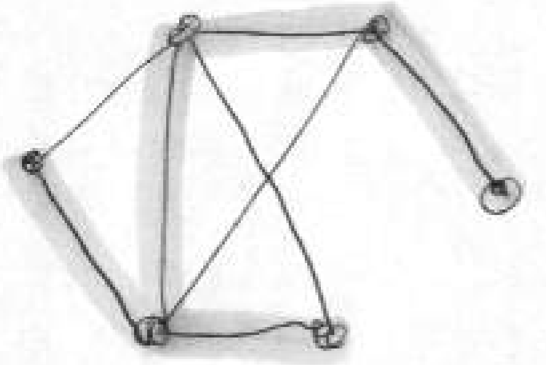
Examples: Consider the network:

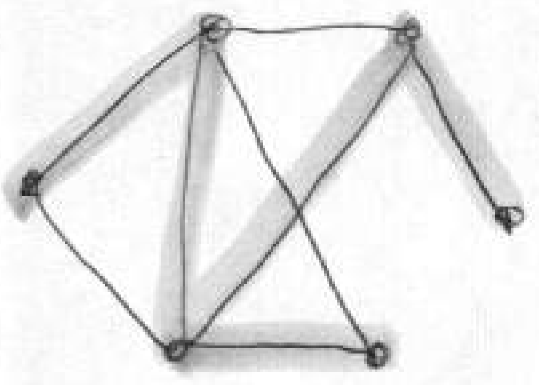


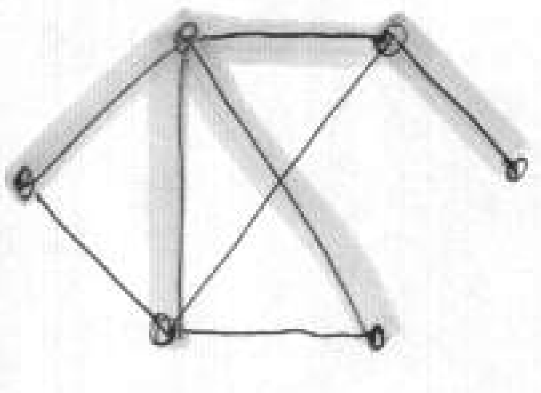
Let's find the spanning trees inside this network.

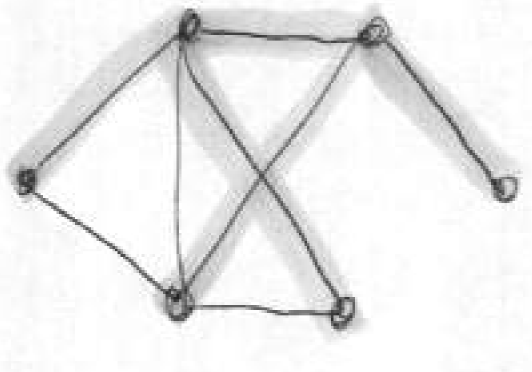


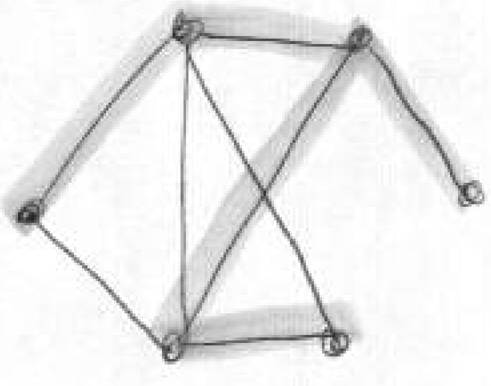


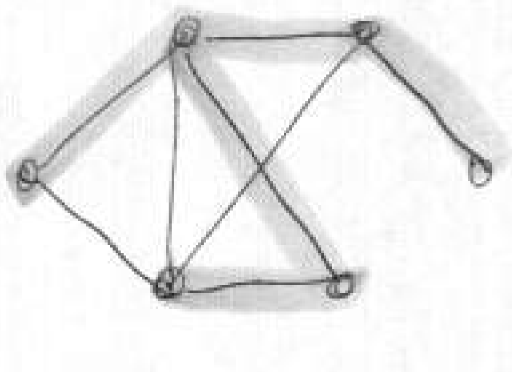












there are even more!

We see that the number of spanning trees in a given network may be very large.

When we have a weighted network, like in the problem about laying internet cable, we can ask for a minimum spanning tree (MST), a spanning tree with the least possible total weight/cost.

The problem we posed earlier can be rephrased as: find an MST in the given weighted graph.

We will do this next time using "Kruskal's Algorithm".