Spanning Trees

A spanning tree for a graph G is a subgraph that is a tree and that contains every vertex.

A minimal spanning tree for a weighted graph is a spanning tree of least weight.

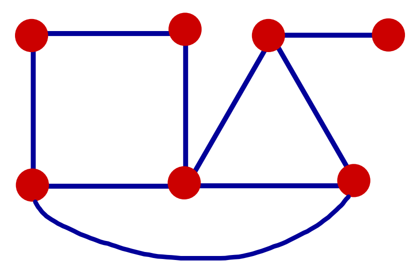
Application: Given a network of roads, which roads should you pave so that (a) all towns are connected and (b) we use the least amount of asphalt?

Spanning Trees

How to find a spanning tree?

One answer: Delete all edges until there are no cycles.

Example. How many spanning trees can you find?

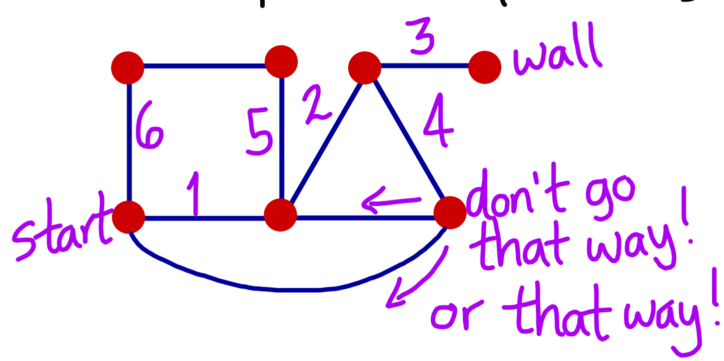


Question. How to find all spanning trees? How many are there?

Could hunt for cycles, delete edges. Inefficient!

Depth-First Search and Breadth-First Search

Depth-first: Start at some point in the graph.
Draw a long path, go as far as possible.
When you hit a wall (= degree 1 vertex),
or an edge that creates a cycle with your
path, back up one step and go in a new direction.



Breadth-first: Use as many edges from start point as
possible. Then from the endpoints of all
those edges use as many edges as possible, etc.

MAZES

One algorithm for solving a maze is to put your right hand on the wall and walk.

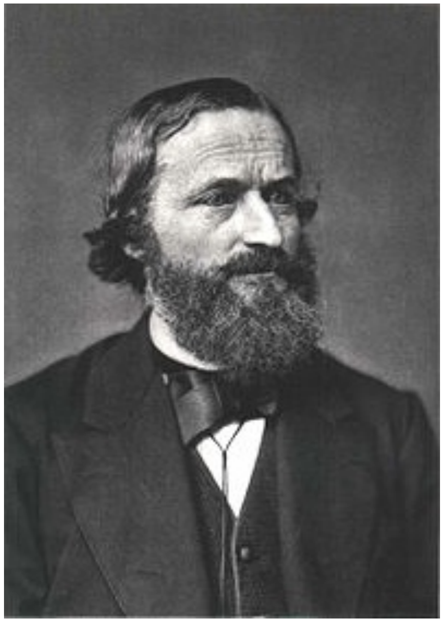
Is this a depth-first or breadth-first algorithm?

Kirchhoff's Theorem

Given a graph with vertices $v_1, \ldots, v_n$, make a matrix M with (i,i)-entry the degree of v_i and all other (i,j)-entries given by: -1 if v_iv_j is an edge
0 otherwise

Theorem. Given a graph G, make the matrix M as above. Delete the i^{th} row and the j^{th} column to obtain a matrix M'. Then:

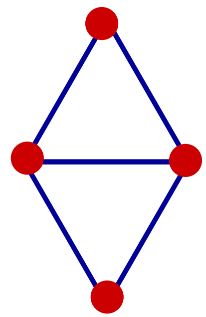
$$(-1)^{i+j} \det(M') = \#\text{ spanning trees for } G.$$



Gustav Kirchhoff

Kirchhoff's Theorem

Example.



Krushkal's Algorithm

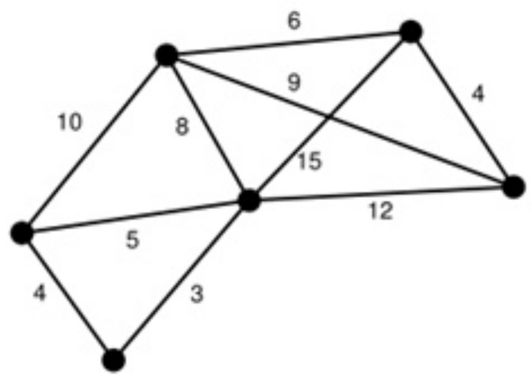
The Algorithm. Set $T = \emptyset$.
Consider all edges e so $T \cup \{e\}$ has no circuits.
Choose the edge e of smallest weight with this property.
Replace T with $T \cup \{e\}$.
Repeat until T is a spanning tree.

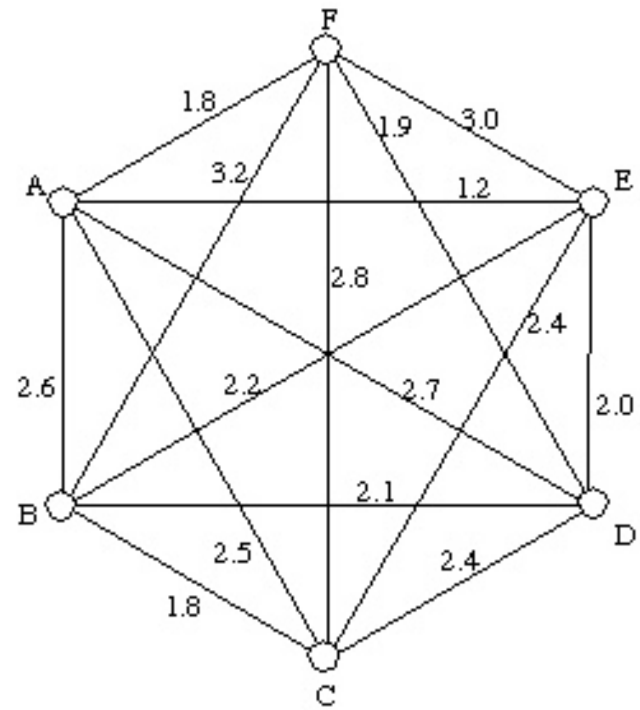
Note: The number of steps is one less than the # of vertices.

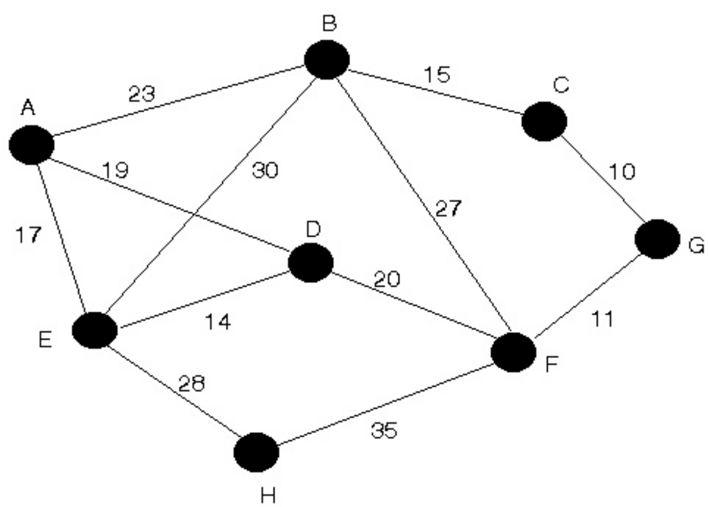
Krushkal's algorithm is an example of a "greedy algorithm"

KRUSHKAL'S ALGORITHM

Find minimal spanning trees for the following weighted graphs.







Krushkal's Algorithm

Why does the algorithm work?

Let $e_1, \ldots, e_{n-1}$ be the edges chosen by Krushkal's algorithm, in order.

Prove the following statement by induction:

$\{e_1, \ldots, e_k\}$ is contained in some minimal spanning tree.

Prim's Algorithm

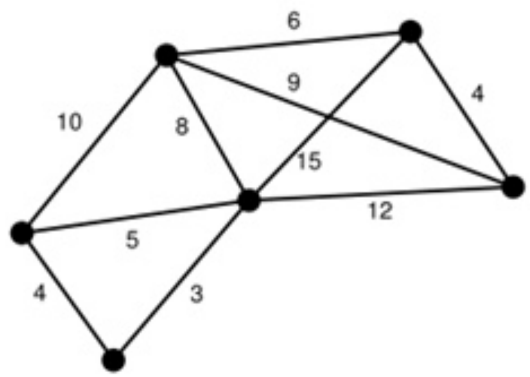
Idea: Grow a tree from a vertex.

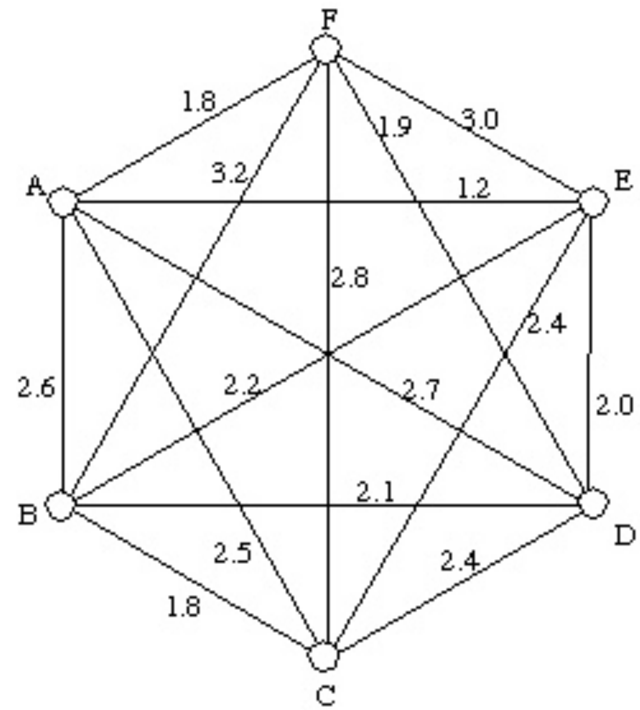
The algorithm. Set $T = v$ (any vertex)
Choose an edge e of minimal weight so $T \cup \{e\}$ is a tree
Replace T with $T \cup \{e\}$.
Repeat until T is a spanning tree.

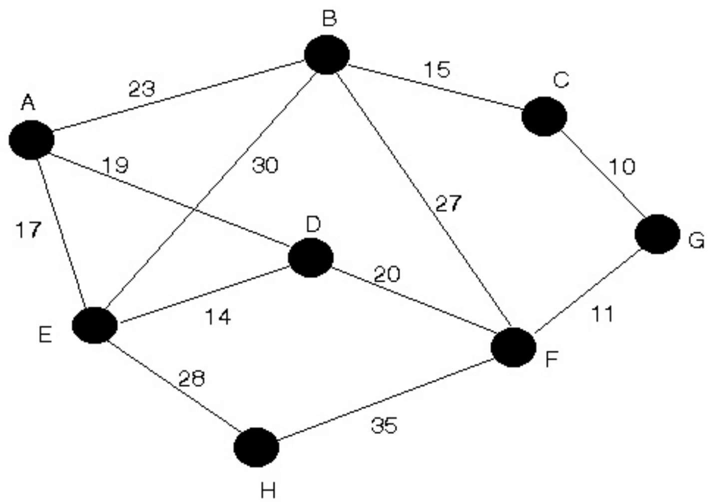
Note: We know $T \cup \{e\}$ is a tree if $T \cap e$ is a single vertex.

Prim's Algorithm

Find minimal spanning trees for the following weighted graphs.







Krushkal's Algorithm vs. Prim's Algorithm

What is the complexity?

size = # edges
cost = # comparisons

Krushkal: $O(n \log n + n^2)$

Prim: $O(n^2)$