Networks, Spanning Trees and Steiner Points

- **Network**

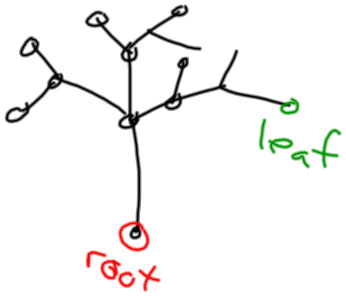
  Another name for a connected graph.
- **Tree**

  A network with no circuits.

  1) connected
  2) No circuits
- **Spanning Tree**

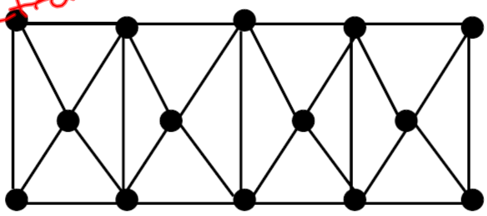
  A *subgraph* that connects all the vertices of the network and has no circuits.
- **Minimum Spanning Tree (MST)**

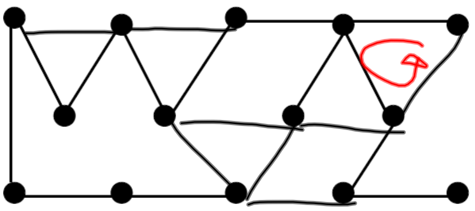
  Among all spanning trees of a *weighted network*, one with the least total weight.



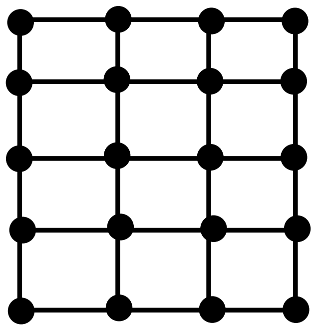
Connected

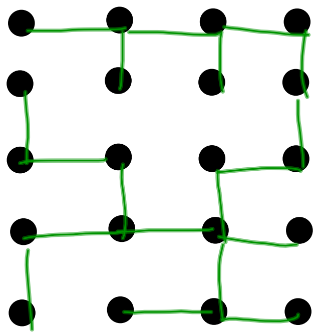
Not a tree





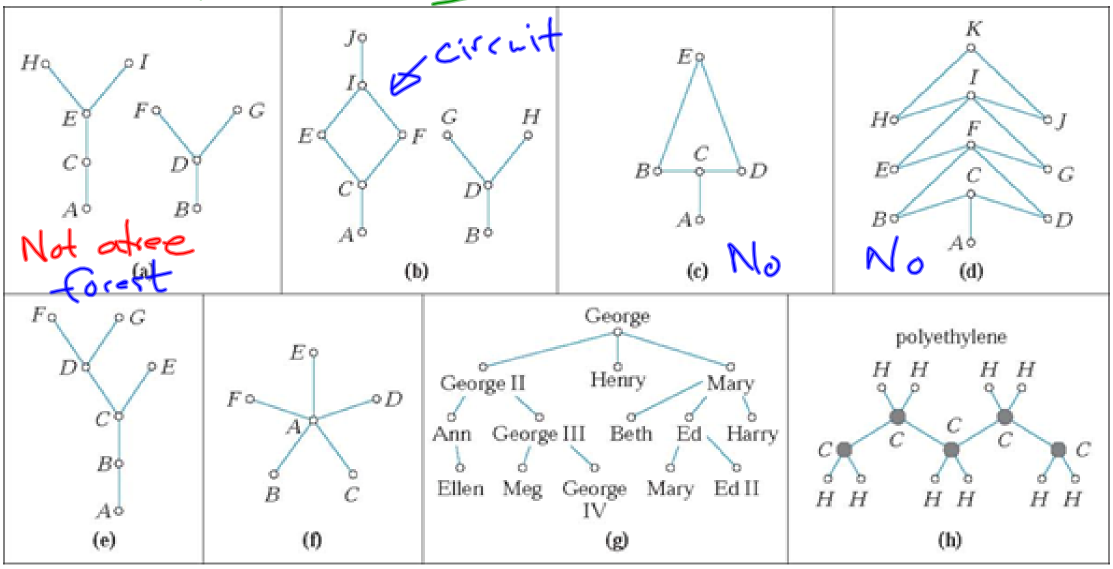
Tree is "minimum connected graph"





any other edge will make a circuit.

Tree is Connected and No circuits



Yes Yes Yes Yes

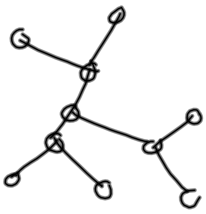
Properties of Trees

Property 1

In a tree, there is one and only one path joining any two vertices.
If there is one and only one path joining any two vertices of a graph, then the graph must be a tree.

Assume Network is Connected and has no circuits ⟹ is a tree

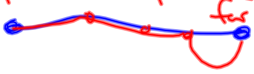
Show there is one and only one path between any two vertices.

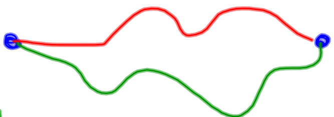


Connected means at least one path.

If there were more than 1 path from a to b.

Paths might be same for awhile





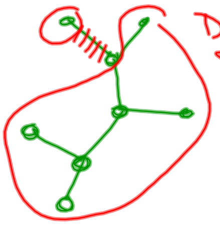
then there is a circuit so it isn't a tree.

A bridge is an edge in a connected graph so that if you remove it the graph falls into 2 components.

Property 2

In a tree, every edge is a bridge.

If every edge of a graph is a bridge, then the graph must be a tree.



Disconnects single vertex from rest

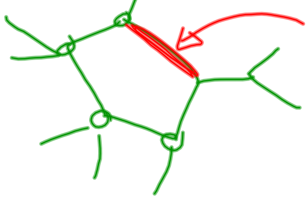
If every edge is a bridge why is it a tree?

why is it connected?

Definition of bridge means Graph Connected

and why doesn't it have circuits?

If it had a circuit

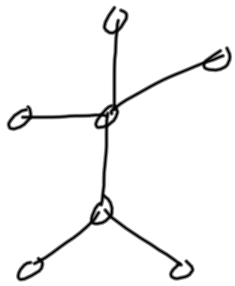


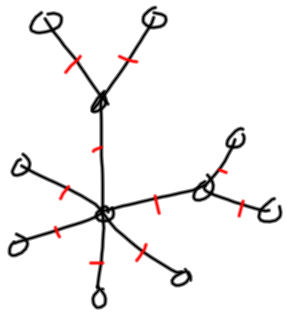
Not a bridge

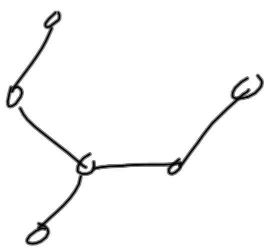
Here do we mean "graphs" or "networks"?

G might not be connected

N always connected







N = # vertices = 7

E = # edges = 6

N = 11

E = 10

N = 6

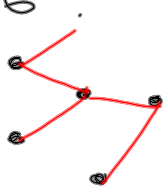
E = 5

Conjecture: In any tree the number of Edges is one less than number of vertices.

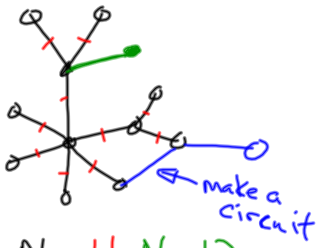
Property 3
A tree with N vertices has $N - 1$ edges.
If a network has N vertices and $N - 1$ edges, then it must be a tree.

(This second statement isn't quite true for all graphs. Just for networks. What is the difference?)

N=6

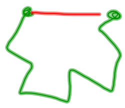


Draw 5 edges
If no circuits
then it is connected.

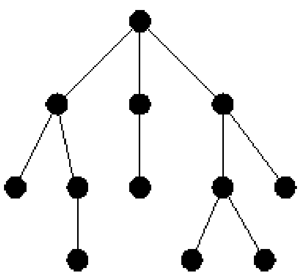


N = 11 N=12
E = 10 N= 11

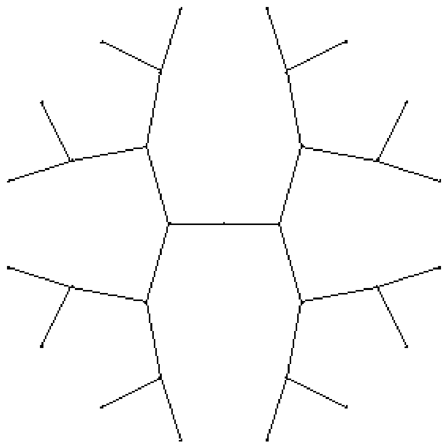
Add an edge (without new vertex)
always makes a circuit.
Because the 2 ends of the edge
were already connected by
a path in the tree.



which makes a
circuit.



N= vertices 12
E= edges 11



N= vertices 30
E= edges 29

Note that every time you add a new vertex and an edge connecting it to the original tree, then you get a new tree and the number of edges is still one less than the number of vertices.

Spanning Trees

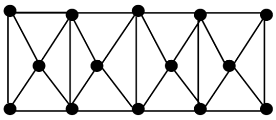
Property 4

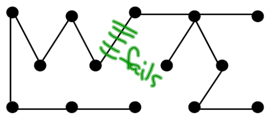
If a network has N vertices and M edges, then $M \geq N - 1$.

$R = M - (N - 1)$ is the redundancy of the network. The number of edges you need to remove to make it a tree.

If M = N – 1, the network is a tree; if M > N –1, the network has circuits and is not a tree. (In other words, a tree is a network with zero redundancy and a network with positive redundancy is not a tree.)

Minimal connected Network





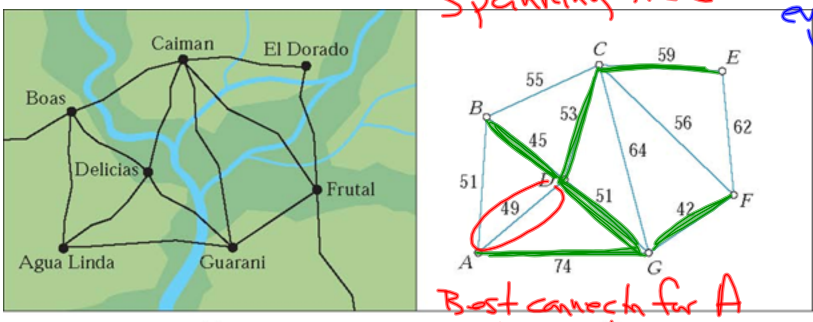
N= 14 Vertices
E= 29 edges
R= 29-(14-1)=16 extra edges

29 edges
13 needed for tree
16 redundant edges

How many edges you need to remove before No cycles left and still connected.

Consider the following map of seven Towns. Suppose your goal is to connect them all with a water irrigation pipeline. The graph at the right shows the "cost" of installing each of the possible pipes.

How would you do it? Cheapest Possible would be a tree. Spanning tree connects every vertex



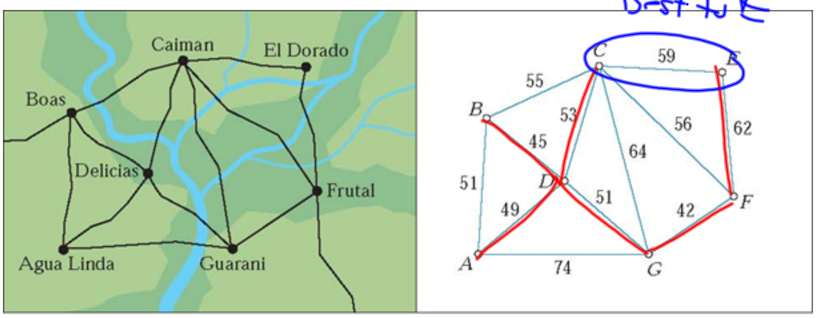
(a) (b)

Total Cost: 324

Best connection for A to rest

try again.....

Best C E



(a) (b)

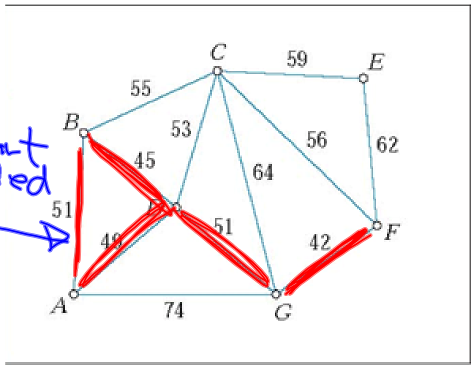
Total Cost: 302

There are many spanning trees in a graph.
How could you find a Minimum Cost Spanning Tree?

Many ways to approach the problem.
We'll discuss "Kruskal's Algorithm".

Add cheapest edges first
w/o making a circuit.

redundant
unneeded



(b)

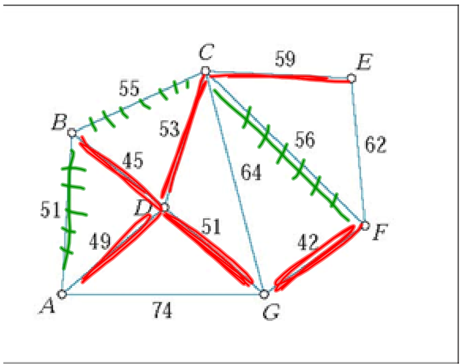
"Kruskal's Algorithm".

First Step. Among all the possible links, we choose the cheapest one,

Each Step: Pick Cheapest possible remaining link so that...

You can't create a circuit

Green edges would make circuits.



(b)

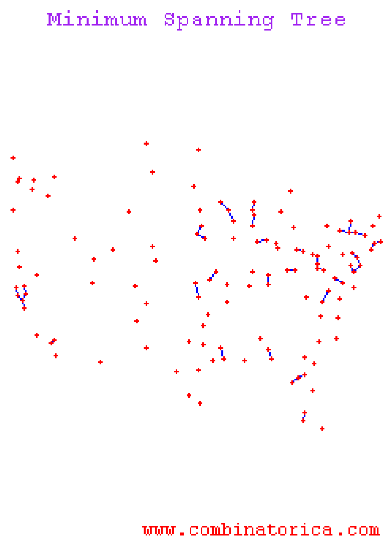
When do you finish?

How many edges do you have to add?

7 vertices ⇒ 6 edges. Done

Two Examples of Kruskal's Algorithm

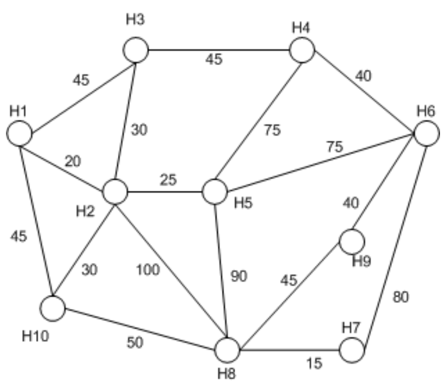
http://www.cut-the-knot.org/Curriculum/Combinatorics/WGraphs.shtml

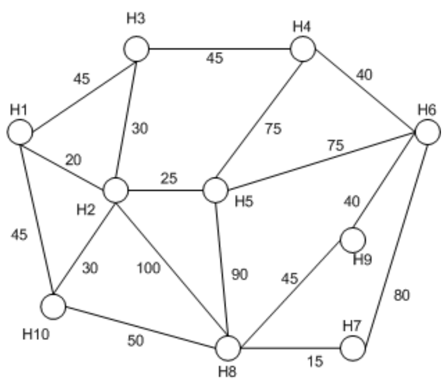


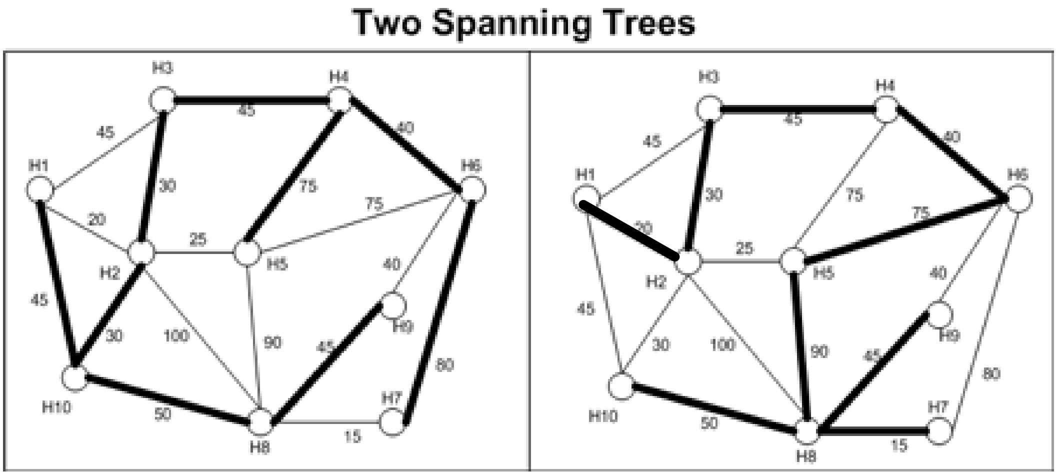
http://students.ceid.upatras.gr/~papagel/project/kruskal.htm

Is Kruskal's Algorithm Optimal?

Does it always give the best possible spanning tree?







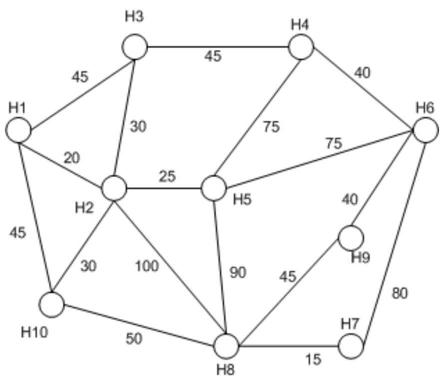
Cost = 45+30+50+30+45+75+40+80+45
= 440

Cost = 20+30+45+40+75+90+50+45+15
= 410

Optimal Spanning Tree has cost 295. How can you improve these spanning trees?

Look at a "cut set of edges" that breaks the graph into two parts.

Try Kruskal's Algorithm



Explanation that shows that Kruskal's Algorithm will always find the optimal spanning tree.

Every edge in the spanning tree is a bridge of the spanning tree.

Removing it breaks the tree into two disconnected parts. There are many edges from one part to the other.

Adding any of them will make a new spanning tree.

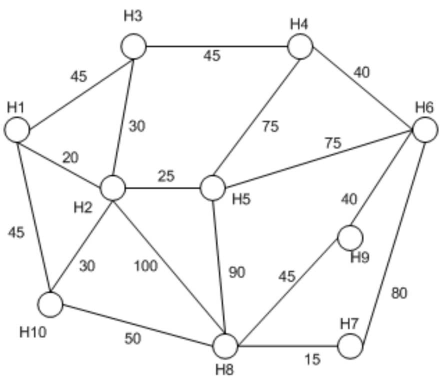
Picking the cheapest edge will make the cheapest of all those spanning trees.

Since Kruskal's algorithm adds the cheapest edges first, this assures that the resulting spanning tree will be the cheapest possible.

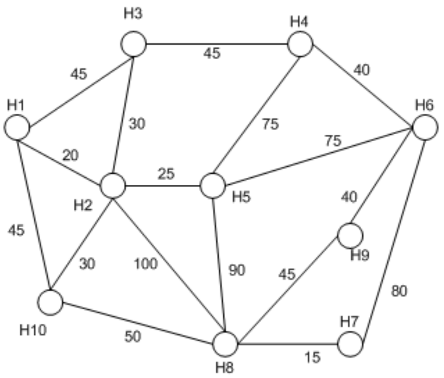
Another algorithm

Prim's Algorithm.

Informally:
1. Pick any point to start.
2. Chose the cheapest edge that connected your "tree-so-far" with a new vertex.
3. Repeat until all the vertices are included.



Start at H1.



Start at H7

Use Prim's Algorithm to generate a maze.

Start with a grid graph.
Give each edge a "random" weight.
Start at bottom right corner.
Add edges following Prim's Algorithm.

http://en.wikipedia.org/wiki/File:MAZE_30x20_Prim.ogv

3. Find all possible Hamilton circuits in the given graph. (Use A as your reference point.)
   - (a) Figure 6-22(a)
   - (b) Figure 6-22(b)
   - (c) Figure 6-22(c)

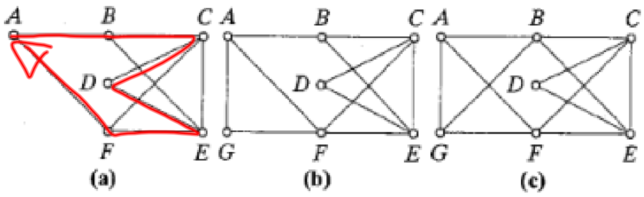
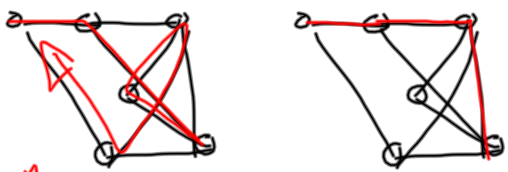


FIGURE 6-22

Theorem: 6 cities have 5! possible Hamilton circuits

Only if "complete graph" all possible edges.



ABEDCF Stuck

29. For the weighted graph shown in Fig. 6-36, (i) find the indicated tour, and (ii) give its cost.
   - (a) An optimal tour (use the brute-force algorithm)
   - (b) The nearest-neighbor tour with starting vertex A
   - (c) The nearest-neighbor tour with starting vertex B
   - (d) The nearest-neighbor tour with starting vertex C

3! = 6 possible routes.

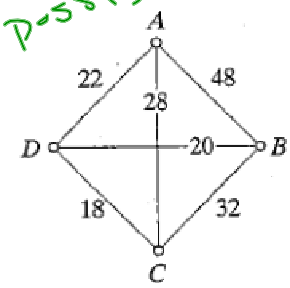
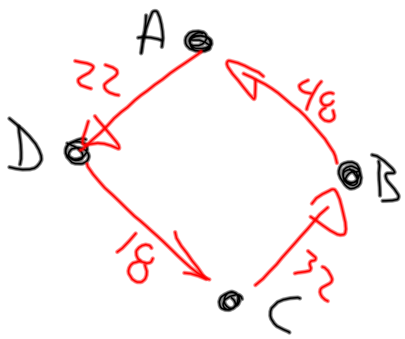


FIGURE 6-36

Try to find cheapest circuit through all cities (vertices)

Nearest Neighbor



Attachments

SingleSharePasswordProtectedVisualCryptographyViaCellularAut.cdf