## Traversability

Traversability is the ability to go along each edge of the network once and only once.

The number of edges into each vertex are counted. If the number is odd, the vertex is said to be "odd".

- A network with no odd vertices is traversable.

  For a network with only even vertices, any vertex may be the beginning point. That same vertex must be the ending point.

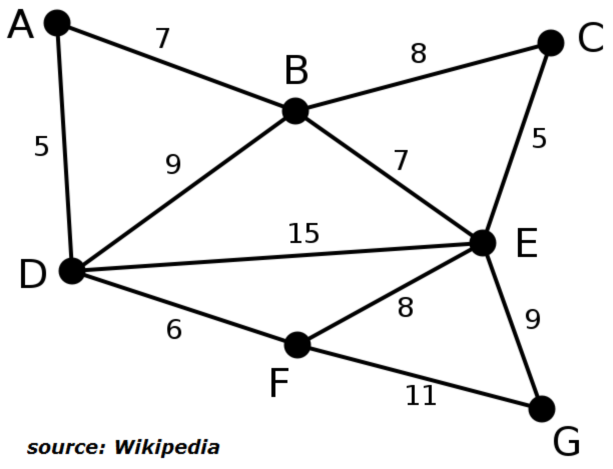
- A network with exactly two odd vertices is traversable.

  For a network with only two odd vertices, either odd vertex may be the beginning point. The other odd vertex must be the ending point.

- A network with more than two odd vertices is not traversable.

(Note: Traversability says nothing about the numbers of times a vertex is visited. The problem about visiting each vertex once and only once is not examined.)

Take the network below.



The vertices A, C and G have two edges.

The vertex F has three edges.

The vertices B and D have four edges.

The vertex E has five edges.

There are two odd vertices – E and F. Therefore the network is traversable, but any path that traverses the network must start at one of E or F and finish at the other.

There are multiple ways this can be done, e.g. F – G – E – C – B – A – D – B – E – D – F – E.

## Shortest Path

While there are algorithms to find the shortest path between two points (e.g. Dijkstra's algorithm) they are more complicated than required for our purposes (they assume computers, that must do everything without being able to exclude obviously wrong paths).

For this unit:

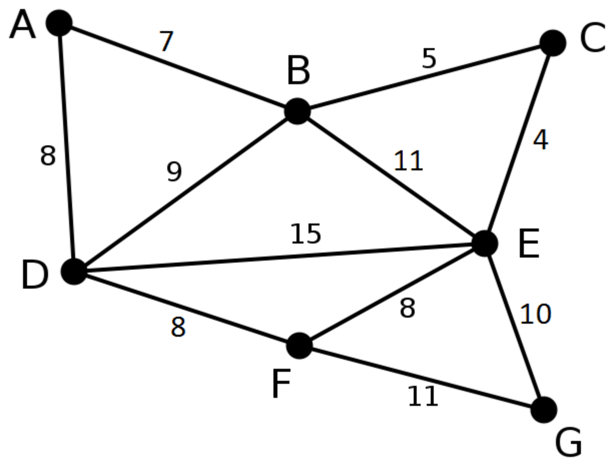
- Find what looks like the shortest path. Carefully note that distance.
- Try other options that might work. Exclude those that are clearly much worse but keep all the others as options.
- Once you have found what you think might be the shortest route or routes, try various small improvements to see if there are shortcuts you may have missed.

Notes:

The shortest route is entirely down to the distances marked on the edges. The number of vertices visited is irrelevant. Do not exclude a path merely because it passes through a lot of vertices.

If the network has one or two "choke" points that must be passed through, then up to these vertices can be dealt with separately, rather than trying to do the full route each time.

Take the network below.



Trying to find the best path from A to G:

A – D – F – G = 27 and A – B – E – G = 28 look the most obvious. The first is shorter, but not by enough to exclude the second.

Examination shows that A – D – E – G is possible, but too long (33).

Careful examination of possible side routes shows that B – C – E is shorter than B – E directly. A – B – C – E – G is 26.

All other possible "detours", e.g. E – F – G rather than E – G directly, add distance to a route.

A – B – C – E – G is the shortest way from A to G.

## Prim's Algorithm

Prim's Algorithm finds the "**minimum spanning tree**" for a network – the edges which connect every point at least once in the least total distance.

- Pick any single vertex.
- Find the vertex directly connected to that vertex with the minimum value.
- With the tree now formed, find the next vertex directly connected (but not already in the tree) with the minimum value.
- Continue in this way until every point has been connected.

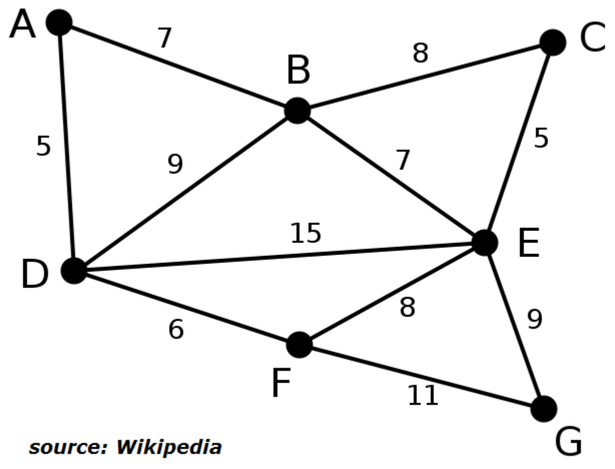
Notes:

While using the algorithm each new branch is selected from anywhere in the partly built network. (Students tend to focus too heavily on the most recently added vertex.)

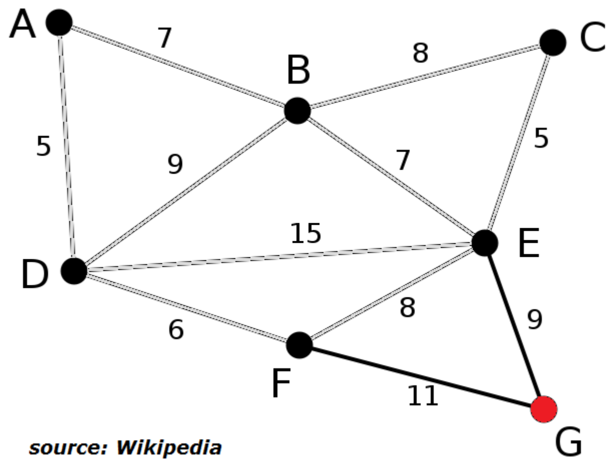
The distance is given by a value on the edge (or elsewhere) and is unrelated to the distance physically shown in diagram.

No loop should be possible in the resulting network. If a loop is present then at least one surplus edge has been added.

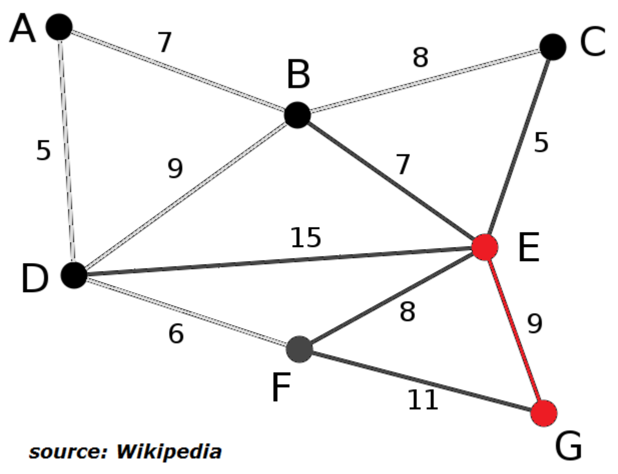
For example, take the network below.



If one starts with G, then there are two possible vertexes to connect to: E and F.

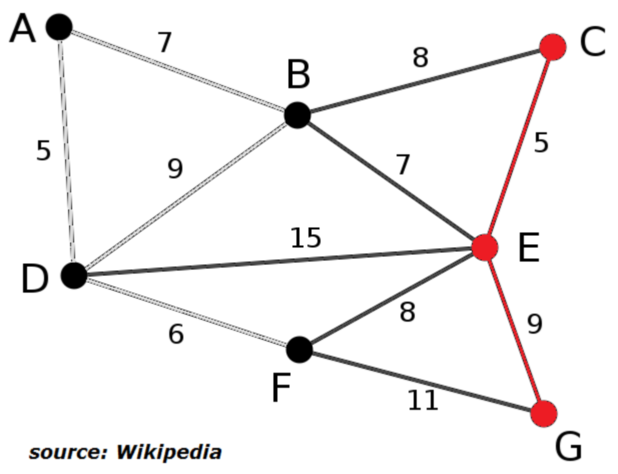


Of these, the closest vertex to G is E. So that is added to the tree.



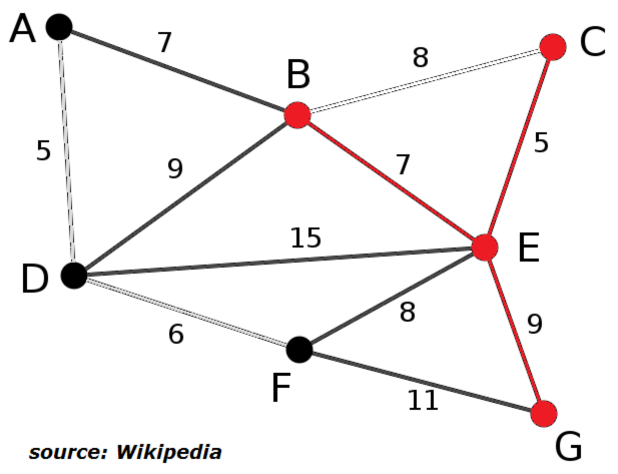
Now with E–G as our tree, we have B, C, D and F (twice) as possible next choices.

The shortest one is to C, so we add that to our tree.



Now our network is C–E–G. There are possible connections to B (twice), D and F (twice).

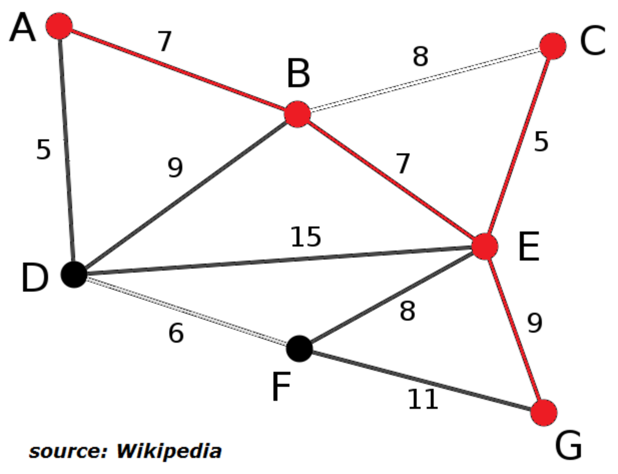
The shortest is E–B, so this is added to the tree.



Our network now is B, C, E and G, and possible connections are to A, D (twice) and F (twice).

The shortest is to A, so this is added to the tree.

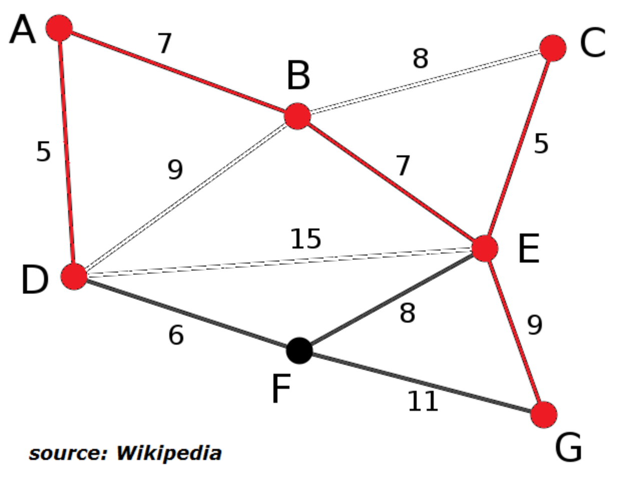
We now have A, B, C, E and G. Only F and D remain outside.



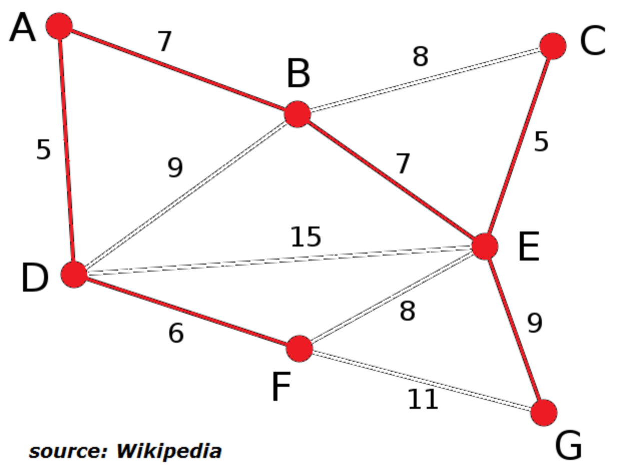
A–D is the shortest of the remaining possible links.

Only F remains outside out tree. Available edges are from D, E and G.

The shortest is to D.



The resulting minimal spanning tree is below, with a value of 39.



A
B
C
D
E
F
G
7
8
5
9
7
5
15
8
9
6
11
source: Wikipedia