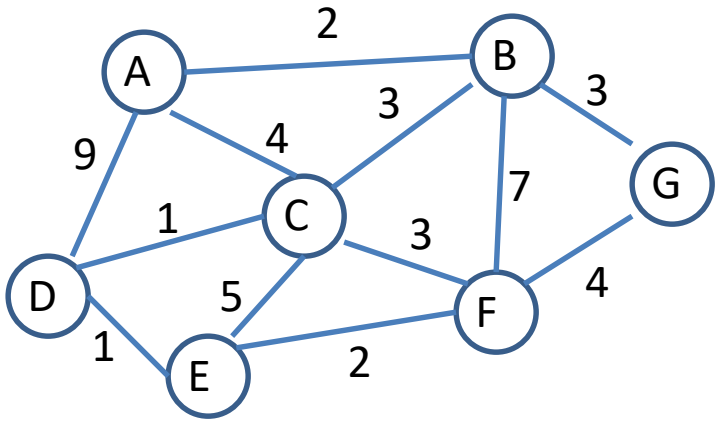
# Prim's
# Minimum Spanning Tree Algorithm

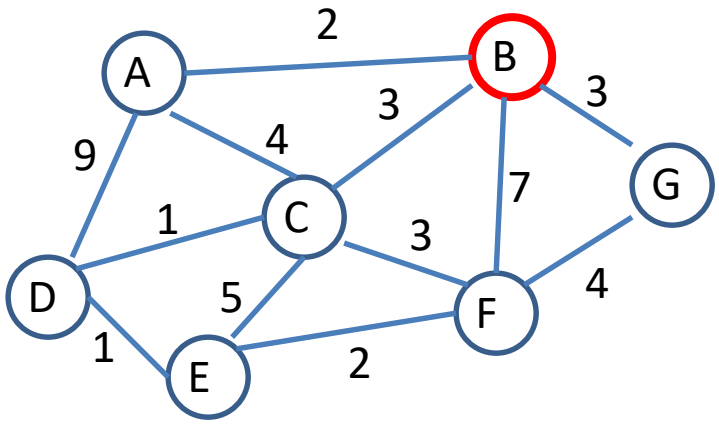
Prim's Algorithm (Robert Prim, 1957, though it was actually discovered by the Czech mathematician Vojtech Jarnik in 1930 and later rediscovered by Edsger Dijkstra in 1960...) starts with a weighted, undirected graph and finds the edges that make up a minimum spanning tree.

Prim's Algorithm: Start with any node. At each step add a node and an edge to the emerging tree. We add the node and the edge to it that is cheapest, among all of the nodes that were not in the current tree. Continue this until all of the nodes have been added. The result must be a tree, for if we have n nodes we have added n-1 edges connecting those nodes; there is no room for a cycle.

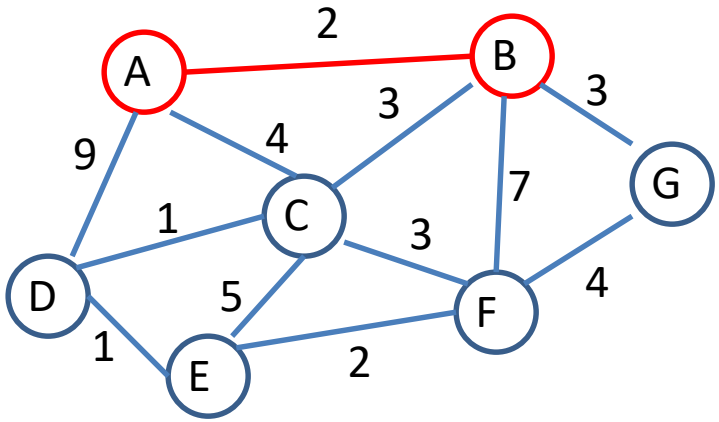
For example:



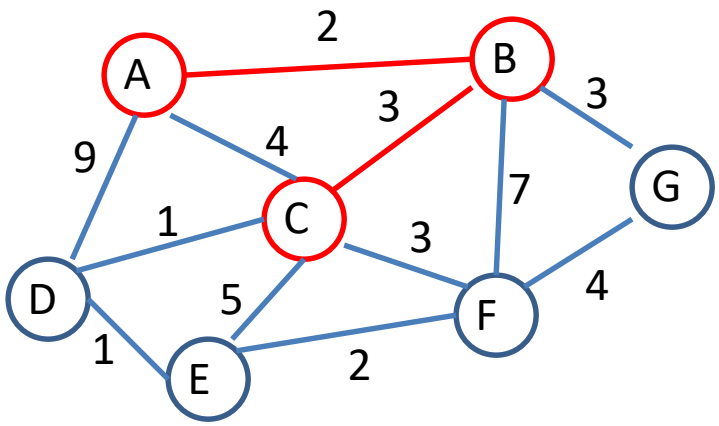

We can start anywhere, so let's start with node B



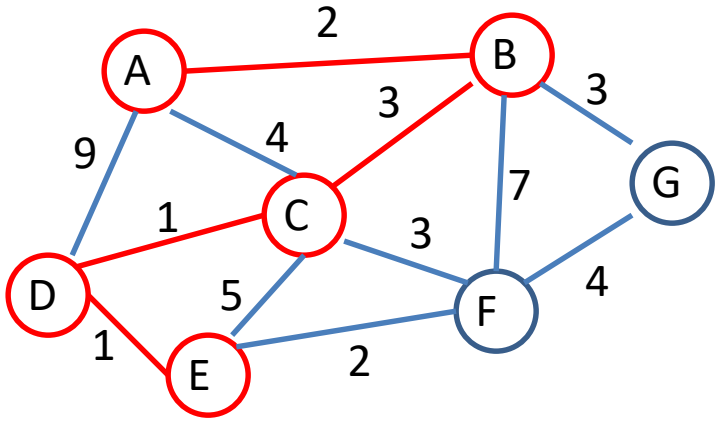

Next we add the cheapest outgoing edge we can from the current tree. This is the edge to A:



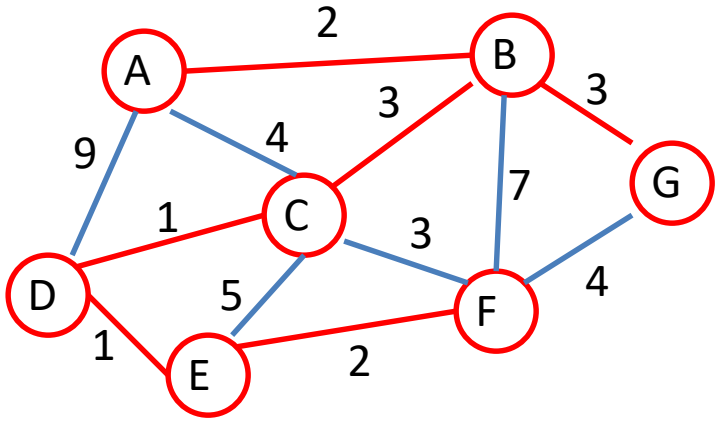

The next cheapest edges are B to C and B to G. Let's try B to C:




Now we have access to some cheap edges. We add C to D, then D to E




The cheapest edge left is E to F. After that we need to add node G; the cheapest way to it is via B

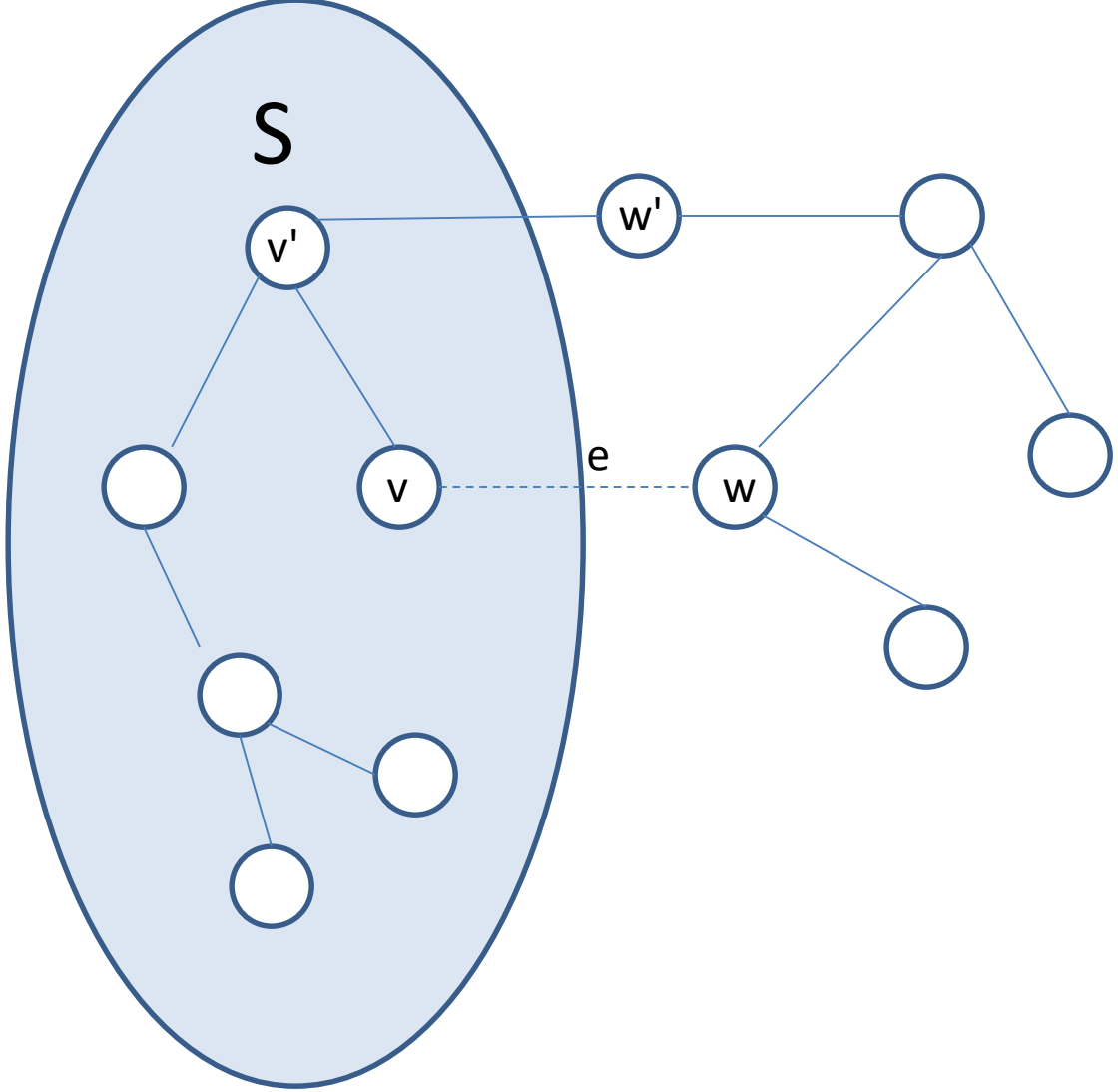



Our minimum cost spanning tree is shown in red.

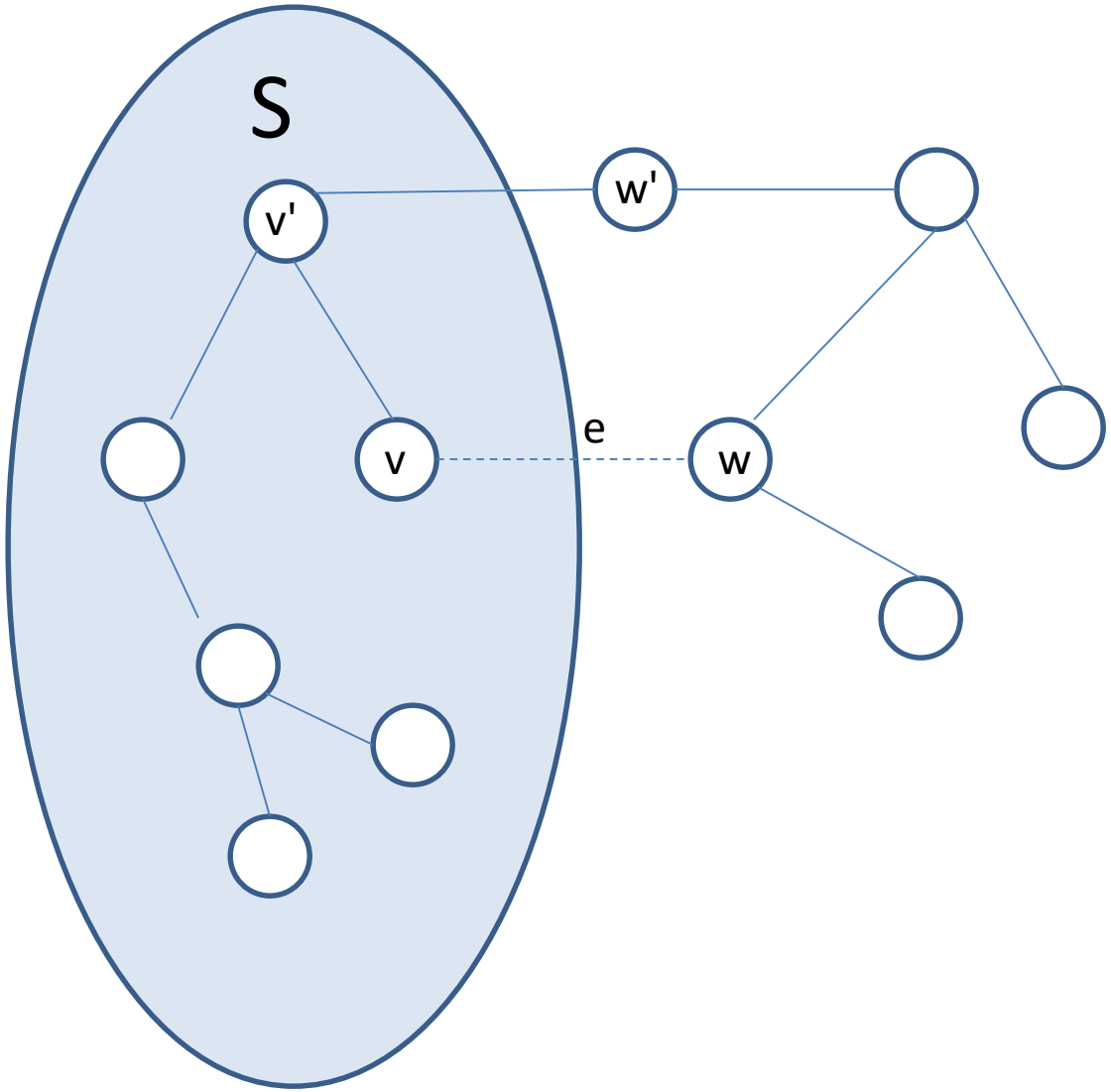
How do we know Prim's algorithm gives a *minimum* spanning tree? The following theorem tells us that it includes the right edges.

Theorem: Let S be any set of nodes in an undirected weighted graph, all of whose edge weights are distinct. Let e, the edge from node v to node w, be the cheapest edge that connects some node in S to a node not in S. Then edge e must be included in any minimum spanning tree for the graph.

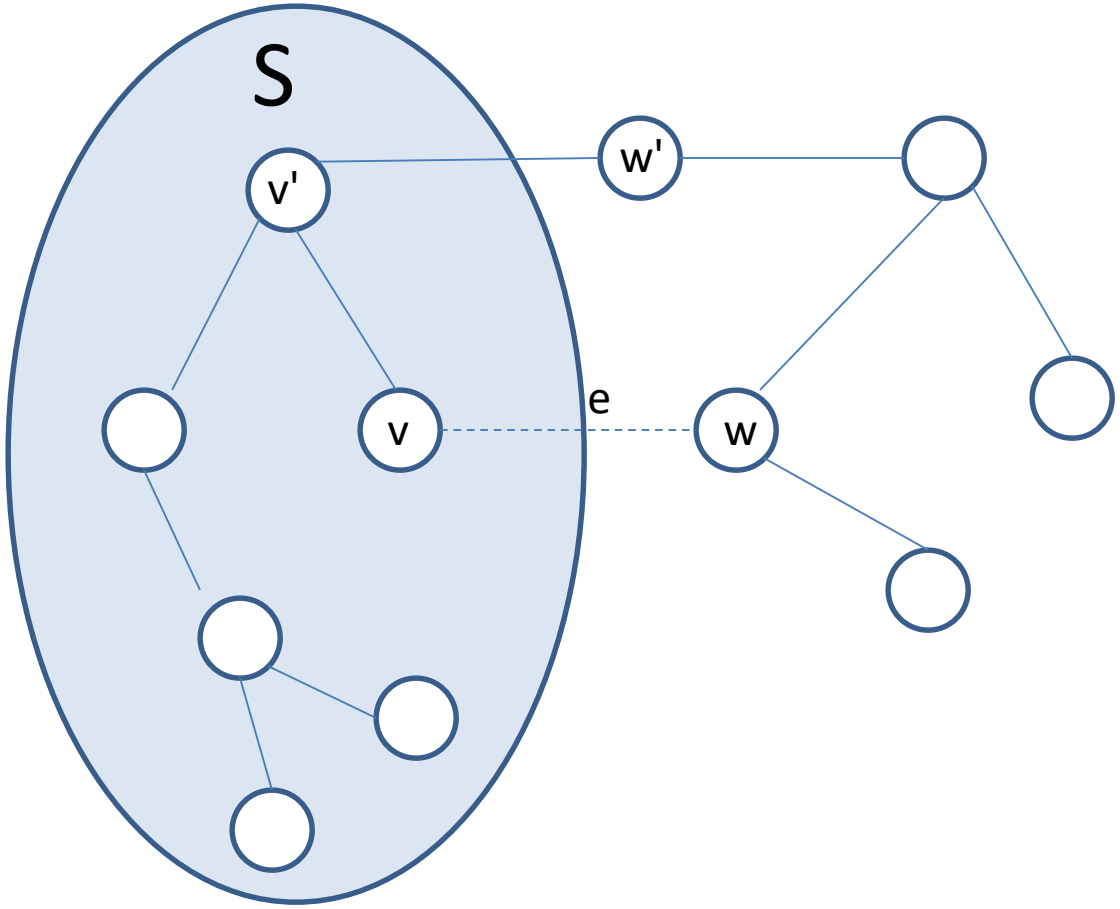
Proof. Here is a picture




Suppose there is a minimum spanning tree that doesn't contain edge e. There must be some path in this tree from node v to node w. Let v' be the last node on this path in S, w' the first node on this path not in S.




Note that edge e must be cheaper than the edge from v' to w', since e is the cheapest edge connecting S to a node not in S.




If we remove the edge from v' to w' and add the edge e, the result will still be a connected set of n-1 edges connecting n nodes, so it is still a spanning tree. And it must be cheaper than the original. This means our original tree could not have been a minimum spanning tree

What if the edge costs aren't all distinct? Let d be the smallest positive difference in any two edge costs. Two spanning trees that have different weights have to differ by at least d. Now take the edges given by Prim's algorithm and lower them by about d/(2*n), jittering them slightly if necessary to make them different. Add an amount around d/(2*n) to other edges. The theorem says that Prim's algorithm gives the minimum spanning tree for the jittered weights. If there was another tree that was minimal for the unjittered weights, its weight would be at least d less than Prim's tree; the jittering affects its weight by at most d/2, so its jittered weight would still be less than Prim's. This contradicts our theorem.

Now, how do we implement Prim's algorithm?

At each step we need to add another edge to our tree; if we look through all of the edges to find the cheapest one that connects a new node to the tree, this takes |E| for each step. Since we add |V|-1 edges, the whole algorithm is O( |V|*|E| ).

There is a better way to organize this. Put all of the nodes in a Priority Queue based on the cost of connecting the node to the current tree. Initially that cost is INFINITY for every node. Each time we remove a node X from the Priority Queue look through its list of outgoing edges and update the cost of adding the destination of each edge to the tree. This means reducing the cost of nodes in the queue, which takes O( log(|V|) ) time. We will eventually cover every node in the graph this way. Altogether, this takes time O( |V| ) to form the originall priority queue, and O( |E|* log( |V|) ) to run the whole algorithm. Altogether, it takes time O( |E|* log( |V|) ) to form the minimum spanning tree.