Section 10.6 – Spanning Trees

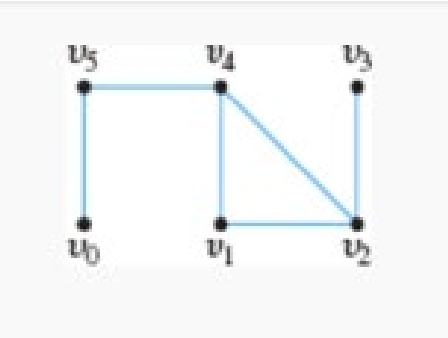
In this section we will discuss Minimum Spanning Trees, but we ***WON'T*** cover Dijkstra's Shortest Path Algorithm.

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

1. Every connected graph has a spanning tree.
2. Any two spanning trees for a graph have the same number of edges.

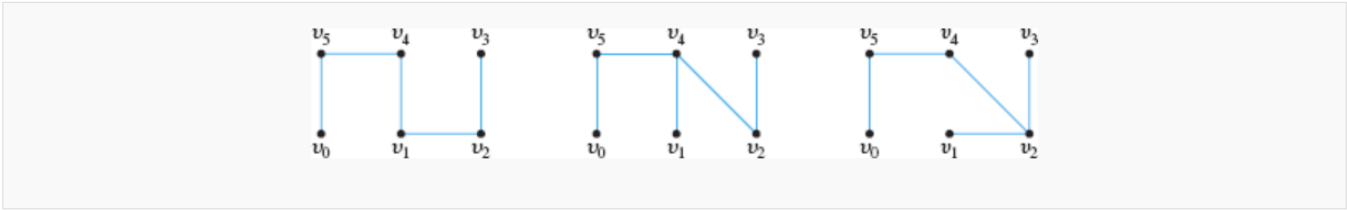
Partner up

Find all spanning trees for the graph G pictured below.



==========

The graph G has one circuit $v_2v_1v_4v_2$, and removing any edge of the circuit gives a tree. Thus, as shown below, there are three spanning trees for G.



A **weighted graph** is a graph for which each edge has an associated positive real number **weight**. The sum of the weights of all the edges is the **total weight** of the graph. A **minimum spanning tree** for a connected, weighted graph is a spanning tree that has the least possible total weight compared to all other spanning trees for the graph.

One solution is to list all spanning trees for the graph, compute the total weight of each, and choose one for which this total is a minimum.

This solution, however, is ***REALLY*** inefficient in its use of computing time because the number of distinct spanning trees is so large.

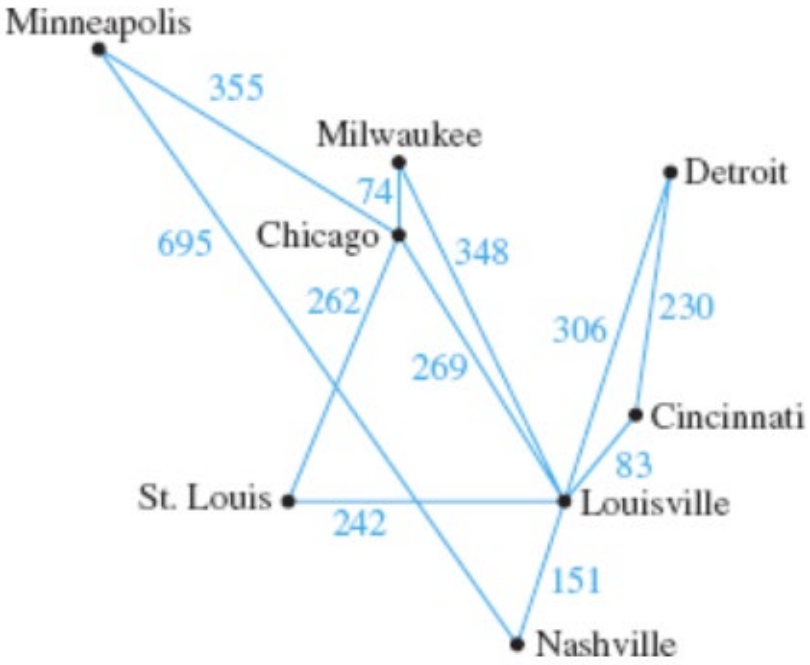
For instance, a complete graph with n vertices has n^{n-2} spanning trees.

Even using the fastest computers available today, examining all such trees in a graph with approximately ***n*** vertices would require more time than is estimated to remain in the life of the universe.

Thankfully, we have two very efficient algorithms to find minimum spanning trees.

Note that these algorithms won't find ***all*** minimum spanning trees of a connected graph, just one of them.

Note also that they are constructive, similar to how we constructed a Hamiltonian Circuit, which also was a sub-graph.



Kruskal's Algorithm

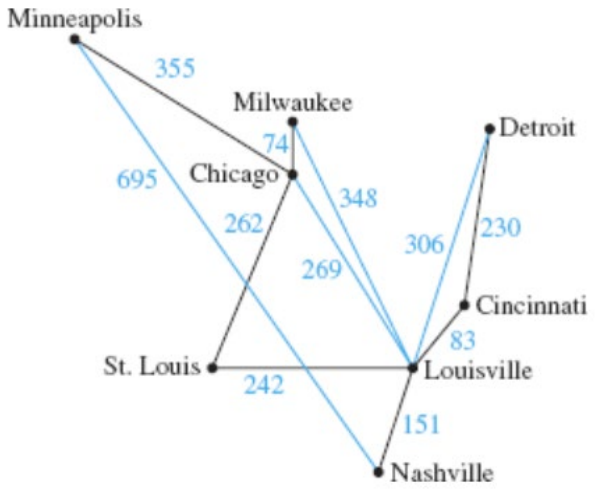
In Kruskal's algorithm, the edges of a connected, weighted graph are examined one by one in order of increasing weight. At each stage the edge being examined is added to what will become the minimum spanning tree, provided that this addition does not create a circuit. After ***n-1*** edges have been added (where ***n*** is the number of vertices of the graph), these edges, together with the vertices of the graph, form a minimum spanning tree for the graph.

Iteration Number	Edge Considered	Weight	Action Taken
1	Chicago–Milwaukee	74	added
2	Louisville–Cincinnati	83	added
3	Louisville–Nashville	151	added
4	Cincinnati–Detroit	230	added
5	St. Louis–Louisville	242	added
6	St. Louis–Chicago	262	added
7	Chicago–Louisville	269	not added
8	Louisville–Detroit	306	not added
9	Louisville–Milwaukee	348	not added
10	Minneapolis–Chicago	355	added

Prim's Algorithm

Prim's algorithm works differently from Kruskal's. It builds a minimum spanning tree ***T*** by expanding outward in connected links from some vertex. One edge and one vertex are added at each stage. The edge added is the one of least weight that connects the vertices already in ***T*** with those not in ***T***, and the vertex is the endpoint of this edge that is not already in ***T*** .

Starting at Minneapolis, construct the minimum spanning tree using Prim's Algorithm.

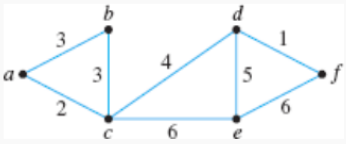


Iteration Number	Vertex Added	Edge Added	Weight
0	Minneapolis		
1	Chicago	Minneapolis–Chicago	355
2	Milwaukee	Chicago–Milwaukee	74
3	St. Louis	Chicago–St. Louis	262
4	Louisville	St. Louis–Louisville	242
5	Cincinnati	Louisville–Cincinnati	83
6	Nashville	Louisville–Nashville	151
7	Detroit	Cincinnati–Detroit	230

Note that the tree obtained is the same as that obtained by Kruskal's algorithm, but the edges are added in a different order.

Partner up

Find all minimum spanning trees for the following graph. Use Kruskal's algorithm and Prim's algorithm starting at vertex a. Indicate the order in which edges are added to form each tree.



==========

Kruskal's adds edges in the following two orders:

1. $\{d, f\}, \{a, c\}, \{a, b\}, \{c, d\}, \{d, e\}$
2. $\{d, f\}, \{a, c\}, \{b, c\}, \{c, d\}, \{d, e\}$

Prim's adds edges in the following two orders:

1. $\{a, c\}, \{a, b\}, \{c, d\}, \{d, f\}, \{d, e\}$
2. $\{a, c\}, \{b, c\}, \{c, d\}, \{d, f\}, \{d, e\}$

The resulting two minimum spanning trees are: