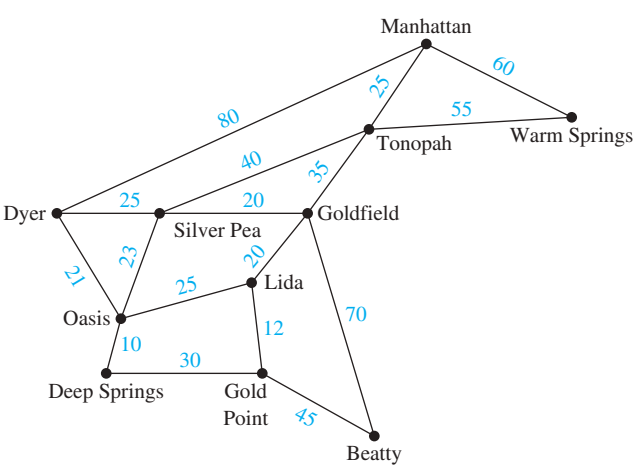
edge in the simple circuit that does not belong to S_{k+1} because S_{k+1} is a tree. By starting at an endpoint of e_{k+1} that is also an endpoint of one of the edges $e_1, \ldots, e_k$, and following the circuit until it reaches an edge not in S_{k+1}, we can find an edge e not in S_{k+1} that has an endpoint that is also an endpoint of one of the edges $e_1, e_2, \ldots, e_k$.

By deleting e from T and adding e_{k+1}, we obtain a tree T' with $n-1$ edges (it is a tree because it has no simple circuits). Note that the tree T' contains $e_1, e_2, \ldots, e_k, e_{k+1}$. Furthermore, because e_{k+1} was chosen by Prim's algorithm at the $(k+1)$st step, and e was also available at that step, the weight of e_{k+1} is less than or equal to the weight of e. From this observation, it follows that T' is also a minimum spanning tree, because the sum of the weights of its edges does not exceed the sum of the weights of the edges of T. This contradicts the choice of k as the maximum integer such that a minimum spanning tree exists containing $e_1, \ldots, e_k$. Hence, $k = n-1$, and $S = T$. It follows that Prim's algorithm produces a minimum spanning tree. ◁

It can be shown (see [CoLeRiSt09]) that to find a minimum spanning tree of a graph with m edges and n vertices, Kruskal's algorithm can be carried out using $O(m \log m)$ operations and Prim's algorithm can be carried out using $O(m \log n)$ operations. Consequently, it is preferable to use Kruskal's algorithm for graphs that are **sparse**, that is, where m is very small compared to $C(n, 2) = n(n-1)/2$, the total number of possible edges in an undirected graph with n vertices. Otherwise, there is little difference in the complexity of these two algorithms.

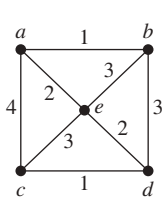
Exercises

1. The roads represented by this graph are all unpaved. The lengths of the roads between pairs of towns are represented by edge weights. Which roads should be paved so that there is a path of paved roads between each pair of towns so that a minimum road length is paved? (*Note:* These towns are in Nevada.)

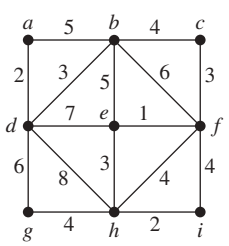


In Exercises 2–4 use Prim's algorithm to find a minimum spanning tree for the given weighted graph.

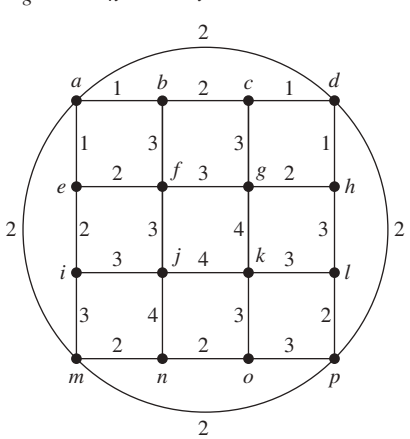
2.



3.



4.



5. Use Kruskal's algorithm to design the communications network described at the beginning of the section.

6. Use Kruskal's algorithm to find a minimum spanning tree for the weighted graph in Exercise 2.

7. Use Kruskal's algorithm to find a minimum spanning tree for the weighted graph in Exercise 3.

8. Use Kruskal's algorithm to find a minimum spanning tree for the weighted graph in Exercise 4.

9. Find a connected weighted simple graph with the fewest edges possible that has more than one minimum spanning tree.

10. A **minimum spanning forest** in a weighted graph is a spanning forest with minimal weight. Explain how Prim's and Kruskal's algorithms can be adapted to construct minimum spanning forests.

A **maximum spanning tree** of a connected weighted undirected graph is a spanning tree with the largest possible weight.

11. Devise an algorithm similar to Prim's algorithm for constructing a maximum spanning tree of a connected weighted graph.

12. Devise an algorithm similar to Kruskal's algorithm for constructing a maximum spanning tree of a connected weighted graph.

13. Find a maximum spanning tree for the weighted graph in Exercise 2.

14. Find a maximum spanning tree for the weighted graph in Exercise 3.

15. Find a maximum spanning tree for the weighted graph in Exercise 4.

16. Find the second least expensive communications network connecting the five computer centers in the problem posed at the beginning of the section.

***17.** Devise an algorithm for finding the second shortest spanning tree in a connected weighted graph.

***18.** Show that an edge with smallest weight in a connected weighted graph must be part of any minimum spanning tree.

19. Show that there is a unique minimum spanning tree in a connected weighted graph if the weights of the edges are all different.

20. Suppose that the computer network connecting the cities in Figure 1 must contain a direct link between New York and Denver. What other links should be included so that there is a link between every two computer centers and the cost is minimized?

21. Find a spanning tree with minimal total weight containing the edges $\{e, i\}$ and $\{g, k\}$ in the weighted graph in Figure 3.

22. Describe an algorithm for finding a spanning tree with minimal weight containing a specified set of edges in a connected weighted undirected simple graph.

23. Express the algorithm devised in Exercise 22 in pseudocode.

Sollin's algorithm produces a minimum spanning tree from a connected weighted simple graph $G = (V, E)$ by successively adding groups of edges. Suppose that the vertices in V are ordered. This produces an ordering of the edges where $\{u_0, v_0\}$ precedes $\{u_1, v_1\}$ if u_0 precedes u_1 or if $u_0 = u_1$ and v_0 precedes v_1. The algorithm begins by simultaneously choosing the edge of least weight incident to each vertex. The first edge in the ordering is taken in the case of ties. This produces a graph with no simple circuits, that is, a forest of trees (Exercise 24 asks for a proof of this fact). Next, simultaneously choose for each tree in the forest the shortest edge between a vertex in this tree and a vertex in a different tree. Again the first edge in the ordering is chosen in the case of ties. (This produces a graph with no simple circuits containing fewer trees than were present before this step; see Exercise 24.) Continue the process of simultaneously adding edges connecting trees until $n - 1$ edges have been chosen. At this stage a minimum spanning tree has been constructed.

***24.** Show that the addition of edges at each stage of Sollin's algorithm produces a forest.

25. Use Sollin's algorithm to produce a minimum spanning tree for the weighted graph shown in

a) Figure 1.
b) Figure 3.

***26.** Express Sollin's algorithm in pseudocode.

****27.** Prove that Sollin's algorithm produces a minimum spanning tree in a connected undirected weighted graph.

***28.** Show that the first step of Sollin's algorithm produces a forest containing at least $\lceil n/2 \rceil$ edges when the input is an undirected graph with n vertices.

***29.** Show that if there are r trees in the forest at some intermediate step of Sollin's algorithm, then at least $\lceil r/2 \rceil$ edges are added by the next iteration of the algorithm.

***30.** Show that when given as input an undirected graph with n vertices, no more than $\lfloor n/2^k \rfloor$ trees remain after the first step of Sollin's algorithm has been carried out and the second step of the algorithm has been carried out $k - 1$ times.

***31.** Show that Sollin's algorithm requires at most $\log n$ iterations to produce a minimum spanning tree from a connected undirected weighted graph with n vertices.

32. Prove that Kruskal's algorithm produces minimum spanning trees.

33. Show that if G is a weighted graph with distinct edge weights, then for every simple circuit of G, the edge of maximum weight in this circuit does not belong to any minimum spanning tree of G.

When Kruskal invented the algorithm that finds minimum spanning trees by adding edges in order of increasing weight as long as they do not form a simple circuit, he also invented another algorithm sometimes called the **reverse-delete algorithm**. This algorithm proceeds by successively deleting edges of maximum weight from a connected graph as long as doing so does not disconnect the graph.

34. Express the reverse-delete algorithm in pseudocode.

35. Prove that the reverse-delete algorithm always produces a minimum spanning tree when given as input a weighted graph with distinct edge weights. [*Hint:* Use Exercise 33.]