Chapter 6

Minimum Spanning Trees

6.1 Introduction

In this chapter, we consider the Minimum Spanning Tree Problem (MSTP). Recall that a spanning tree T of a graph G is a connected acyclic subgraph that spans all the nodes. Every spanning tree of G has $n - 1$ edges. Given an undirected graph $G = (N, E)$ with $|N| = n$ nodes and $|E| = m$ edges with a *length* or *cost* c_{ij} associated with each edge $\{i, j\} \in E$, we wish to find a spanning tree, called a MST, that has the smallest total cost (or length) of its edges, measured as the sum of the costs of the edges in the spanning tree.

6.2 Applications

The MSTP arises in one of two ways, directly or indirectly. In some *direct* applications, we wish to connect a set of points using the least-cost connection of edges. The points represent physical entities such as components of a computer chip or users of a system who must be connected to each other or to a central service such as a central processor in a computer system. In *indirect* applications, we may wish to connect some set of points using a measure of performance.

6.2.1 Designing Physical Systems

The design of physical systems is a complex task involving an interplay between performance objectives (such as throughput and reliability), design costs, operating economics, and available technology. In many settings, the major criterion is simple: we need to design a network that connects geographically dispersed system components or just provides the infrastructure needed for users to communicate with each other. In many of these settings, the system does not need any redundancy, so we are interested in the simplest possible connection, i.e., a spanning tree. This type of application arises in the construction (or installation) of numerous physical systems: highways, computer networks, telephone networks, railroads, cable television lines, and electrical power transmission lines. For example, this type of MSTP arises in the following problem settings:

- Connect terminals in cabling the panels of electrical equipment. How should we wire the terminals to use the least possible length of the wire?

- Constructing a pipeline network to connect a number of towns using the smallest possible total length of pipeline.
- Linking isolated villages in a remote region connected by roads but not yet by telephone service. We wish to determine along which stretches of roads we should place telephone lines, using the minimum possible total miles of the lines, to link every pair of villages.
- Constructing a digital computer system, composed of high-frequency circuitry, when it is important to minimize the length of wires between different components to reduce both capacitance and delay line effects. Since all components must be connected, we face a MSTP.
- Connecting a number of computer sites by high-speed lines. Each line is available for leasing at a certain monthly cost, and we wish to determine a configuration that connects all the sites at the least possible cost.

6.2.2 All-Pairs Minimax Path Problem

In a graph $G = (N, E)$ with edge costs c_{ij}, $\{i, j\} \in E$, we define the value of a path P from node k to node l as the maximum cost edge in P. The all-pairs minimax path problem requires that we determine, for every pair $[k, l]$ of nodes, a minimum value path from k to l. We show how to solve the all-pairs minimax path problem on an undirected graph by solving a single MSTP.

The minimax path problem arises in a variety of situations. As an example, consider a spacecraft that is about to enter the earth's atmosphere. The craft passes through different pressure and temperature zones that we can represent by edges of a graph. It needs to fly along a trajectory that will bring the craft to the surface of the earth while keeping the maximum temperature to which the surface of the craft is exposed as low as possible.

To transform the all-pairs minimax path problem into a MSTP, let T^* be a minimum spanning tree of G. Let P denote the unique path in T^* between a node pair $[p, q]$, and let $\{i, j\}$ denote the maximum cost edge in P. Observe that the value of the path P is c_{ij}. By deleting edge $\{i, j\}$ from T^*, we partition the node set N into two subsets and therefore define an $i - j$ cut $(S, \overline{S})$ with $i \in S$ and $j \in \overline{S}$ (see Figure 6.1). We later show that this cut satisfies the following property:

$$c_{ij} \leq c_{kl} \quad \forall \{k, l\} \in E \ : \ k \in S, l \in \overline{S} \tag{6.1}$$

for otherwise by replacing the edge $\{i, j\}$ by an edge $\{k, l\}$, we can obtain a spanning tree of smaller cost. Now, consider any path P' from node p to node q. This path must contain at least one edge $\{k, l\}$ crossing $p - q$ cut $(S, \overline{S})$. Condition (6.1) implies that the value of the path P' will be at least c_{ij}. Since c_{ij} is the value of the path P, P must be a minimum value path from node p to node q. This observation establishes the fact that the unique path between any pair of nodes in T^* is the minimum value path between that pair of nodes.

6.2.3 Reducing Data Storage

In several different application contexts, we wish to store data specified in the form of a two-dimensional array more efficiently than storing all the elements of the array (to save memory space). We assume that the rows of the array have many similar entries and differ only at a few places.

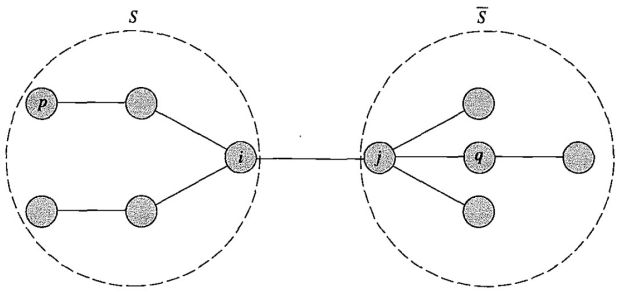


Figure 6.1: Cut $(S, \overline{S})$ formed by deleting the edge $\{i, j\}$ from a spanning tree

Since the entities in the rows are similar, one approach for saving memory is to store one row, called the reference row, completely, and to store only the differences between some of the rows so that we can derive each row from these differences and the reference row. Let c_{ij} denote the number of different entries in rows i and j; that is, if we are given row i, then by making c_{ij} changes to the entries in this row we can obtain row j, and vice versa. Suppose that the array contains four rows, represented by R_1, R_2, R_3, and R_4, and we decide to treat R_1 as a reference row. Then one plausible solution is to store the differences between R_1 and R_2, R_2 and R_4, and R_1 and R_3. Clearly, from this solution, we can obtain rows R_2 and R_3 by making c_{12} and c_{13} changes to the elements in row R_1. Having obtained row R_2, we can make c_{24} changes to the elements of this row to obtain R_4.

It is sufficient to store differences between those rows that correspond to edges of a spanning tree. These differences permit us to obtain each row from the reference row. The total storage requirement for a particular storage scheme is the length of the reference row plus the sum of the differences between the rows. Therefore, a MST provides a least-cost storage scheme.

6.2.4 Cluster Analysis

The essential issue in cluster analysis is to partition a set of data into *natural groups*; the data points within a particular group of data, or a *cluster*, should be more closely related to each other than the data points not in that cluster.

Cluster analysis is important in a variety of disciplines that rely on empirical investigations. Consider, for example, an instance of a cluster analysis arising in medicine. Suppose that we have data on a set of 350 patients, measured with respect to 18 symptoms. Suppose, further, that a doctor has diagnosed all of these patients as having the same disease, which is not well understood. The doctor would like to know if he can develop a better understanding of this disease by categorizing the symptoms into smaller groupings that can be detected through cluster analysis. Doing so might permit the doctor to find more natural disease categories to replace or subdivide the original disease.

We can solve a class of problems arising in cluster analysis by solving a MSTP. Suppose that we are interested in finding a partition of a set of n points in two-dimensional Euclidean space into clusters. A popular method for solving this problem is finding a MST and obtain k partitions by starting with a MST and delete k arcs from the MST one by one in non-increasing order of their lengths.

We illustrate the latter approach using an example. Consider a set of 27 points shown in Figure 6.2(a). Suppose that the network in Figure 6.2(b) is a MST for these points. Deleting the three largest length arcs from the MST gives a partition with four clusters shown in Figure 6.2(c).

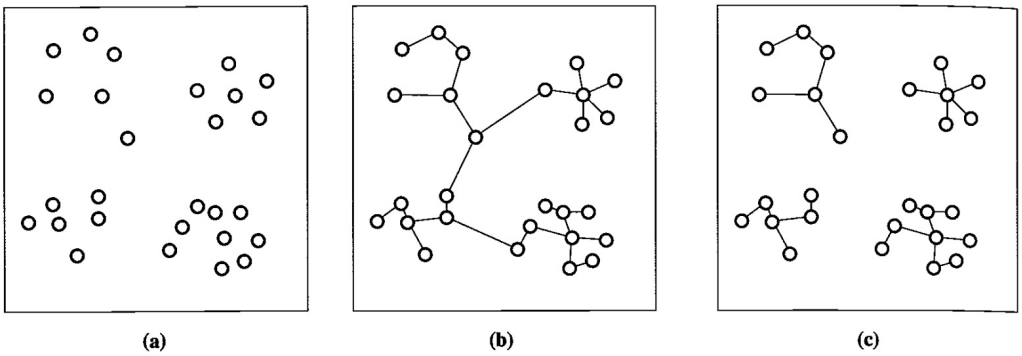


Figure 6.2: Identifying clusters by finding a minimum spanning tree

6.3 Optimality Conditions

For the MSTP, we can formulate the optimality conditions in two equivalent ways: *cut optimality conditions* and *path optimality conditions*.

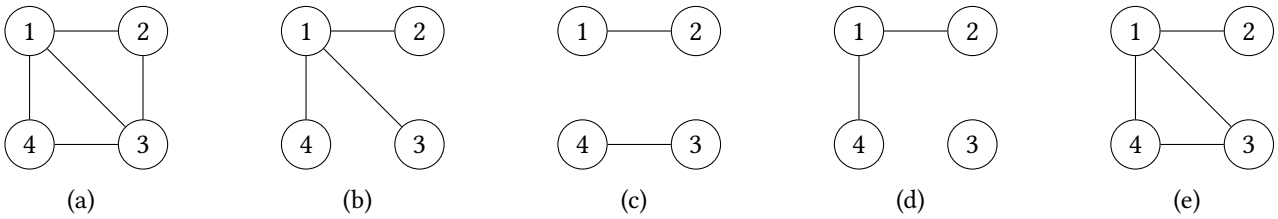


Figure 6.3: A graph (a) and a corresponding spanning tree (b), along with three non-spanning trees (a disconnected one (c), one that doesn't span all nodes (d), and a cyclic subgraph (e))

Before considering these optimality conditions, let us establish some further notation and illustrate some basic concepts. Figure 6.3(a) shows a graph with four nodes. The subgraph 6.3(b) is a spanning tree. The other three subgraphs 6.3(c), 6.3(d), 6.3(e) are non-spanning tree because (c) is not connected, (d) does not span all nodes, and (e) contains a cycle. We refer to those edges contained in a given spanning tree as *tree edges* (i.e., edges $\{1,2\}$, $\{1,3\}$, and $\{1,4\}$ in Figure 6.3(b)) and to those edges not contained in a given spanning tree as *non-tree edges* (i.e., edges $\{2,3\}$ and $\{3,4\}$ in Figure 6.3(b)).

The following two elementary observations will arise frequently in our development in this chapter.

1. For every non-tree edge $\{k,l\}$, the spanning tree T contains a unique path from node k to node l. The edge $\{k,l\}$ together with this unique path defines a cycle (see Figure 6.4(a)).

2. If we delete any tree edge $\{i,j\}$ from a spanning tree, the resulting graph partitions the node set N into two subsets (see Figure 6.4(b)). The edges from the underlying graph G whose two endpoints belong to the different subsets constitute an $i-j$ cut.

We next prove the two optimality conditions.

Theorem 6.1. ***(Cut Optimality Conditions)*** *A spanning tree T^* is a MST if and only if it satisfies the following cut optimality conditions: for every tree edge $\{i,j\} \in T^*$, $c_{ij} \le c_{kl}$ for every edge $\{k,l\}$ contained in the $i-j$ cut formed by deleting edge $\{i,j\}$ from T^*.*

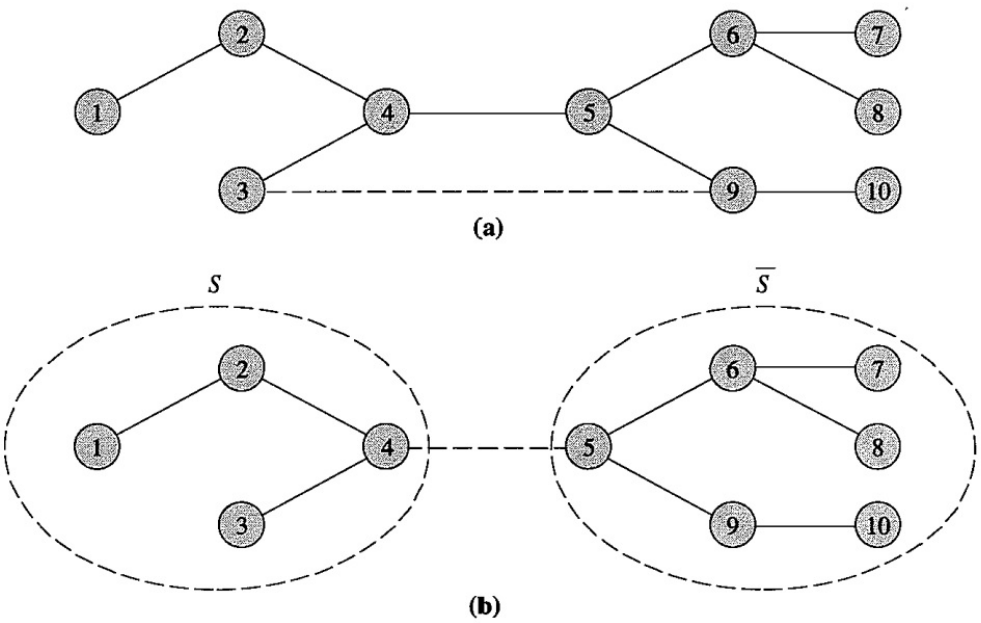


Figure 6.4: Illustrating properties of a spanning tree: (a) adding edge $\{3, 9\}$ to the spanning tree forms the unique cycle $3 - 4 - 5 - 9 - 3$; (b) deleting edge $\{4, 5\}$ forms the $4 - 5$ cut $(S, \overline{S})$ with $S = \{1, 2, 3, 4\}$

Proof. It is easy to see that every MST T^* must satisfy the cut optimality conditions. For, if $c_{ij} > c_{kl}$ and edge $\{k, l\}$ is contained in the $i - j$ cut formed by deleting edge $\{i, j\}$ from T^*, then introducing edge $\{k, l\}$ into T^* in place of edge $\{i, j\}$ would create a spanning tree with a cost less than T^*, contradicting the optimality of T^*.

We next show that if any tree T^* satisfies the cut optimality conditions, it must be optimal. Suppose that T' is a MST and $T' \neq T^*$. Then, T^* contains an edge $\{i, j\}$ that is not in T'. Deleting edge $\{i, j\}$ from T^* creates a cut, say $(S, \overline{S})$. If we add the edge $\{i, j\}$ to T', we create a cycle W that must contain an edge $\{k, l\}$ (other than edge $\{i, j\}$) with $k \in S$ and $l \in \overline{S}$. Since T^* satisfies the cut optimality conditions, then $c_{ij} \leq c_{kl}$. Moreover, since T' is a MST, $c_{ij} \geq c_{kl}$, for otherwise we could improve on its cost by replacing edge $\{k, l\}$ by edge $\{i, j\}$. Therefore, $c_{ij} = c_{kl}$. If we introduce edge $\{k, l\}$ in the tree T^* in place of edge $\{i, j\}$, we produce another MST and it has one more edge in common with T'. Repeating this argument several times, we can transform T^* into the MST T'. This construction shows that T^* is also a MST and completes the proof. ∎

The cut optimality conditions imply that every edge in a MST is a minimum cost edge across the cut that is defined by removing it from the tree.

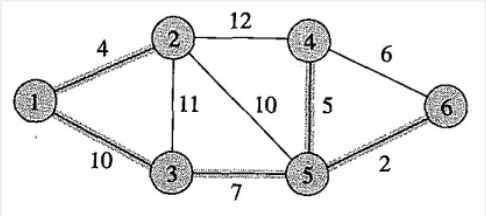
Theorem 6.2. ***(Path Optimality Conditions)*** *A spanning tree T^* is a MST if and only if it satisfies the following path optimality conditions: for every non-tree edge $\{k, l\}$ of G, $c_{ij} \leq c_{kl}$ for every edge $\{i, j\}$ contained in the path in T^* connecting nodes k and l.*

Proof. It is easy to show the necessity of the path optimality conditions. Suppose T^* is a MST satisfying these conditions and edge $\{i, j\}$ is contained in the path in T^* connecting nodes k and l. If $c_{ij} > c_{kl}$, introducing edge $\{k, l\}$ into T^* in place of edge $\{i, j\}$ would create a spanning tree with a cost less than T^*, contradicting the optimality of T^*.

We establish the sufficiency of the path optimality conditions by using the sufficiency of the cut optimality conditions. Let $\{i,j\}$ be any tree edge in T^*, and let S and $\overline{S}$ be the two sets of connected nodes produced by deleting edge $\{i,j\}$ from T^*. Suppose $i \in S$ and $j \in \overline{S}$. Consider any edge $\{k,l\}$ with $k \in S$ and $l \in \overline{S}$. Since T^* contains a unique path joining nodes k and l and edge $\{i,j\}$ is the only edge in T^* joining a node in S and a node in $\overline{S}$, edge $\{i,j\}$ must belong to this path. The path optimality condition implies that $c_{ij} \leq c_{kl}$. Since this condition must be valid for every non-tree edge $\{k,l\}$ in the cut $(S,\overline{S})$ formed by deleting any tree edge $\{i,j\}$, T^* satisfies the cut optimality conditions and must be a MST. ∎

Exercise 6.1.

In the following graph, the bold lines represent a MST



1. By listing each tree edge $\{i,j\}$ and the minimum length edge in the cut defined by the edge $\{i,j\}$, verify that the tree satisfies the cut optimality conditions.
2. By listing each non-tree edge $\{k,l\}$ and the maximum length edge on the tree path from node k to node l, verify that this tree satisfies the path optimality conditions.

1. For each tree edge $\{i,j\}$, we list the edges of the $i-j$ cut $(S,\overline{S})$ (i.e., $E(S,\overline{S})$) and check that edge $\{i,j\}$ has the minimum cost among the edge in the cutset
   - $\{i,j\} = \{1,2\}$, $E(S,\overline{S}) = \{\{1,2\},\{2,3\},\{2,4\},\{2,5\}\}$, $c_{12} = 4$, $c_{23} = 11$, $c_{24} = 12$, $c_{25} = 10$
   - $\{i,j\} = \{1,3\}$, $E(S,\overline{S}) = \{\{1,3\},\{2,3\},\{2,4\},\{2,5\}\}$, $c_{13} = 10$, $c_{23} = 11$, $c_{24} = 12$, $c_{25} = 10$
   - $\{i,j\} = \{3,5\}$, $E(S,\overline{S}) = \{\{2,4\},\{2,5\},\{3,5\}\}$, $c_{24} = 12$, $c_{25} = 10$, $c_{35} = 7$
   - $\{i,j\} = \{4,5\}$, $E(S,\overline{S}) = \{\{2,4\},\{4,5\},\{4,6\}\}$, $c_{24} = 12$, $c_{45} = 5$, $c_{56} = 6$
   - $\{i,j\} = \{5,6\}$, $E(S,\overline{S}) = \{\{4,6\},\{5,6\}\}$, $c_{46} = 6$, $c_{56} = 2$
2. For each non-tree edge $\{k,l\}$, we list the path $P(k,l)$ connecting k and l and the cost of the maximum-cost edge in the path
   - $\{k,l\} = \{2,3\}$, $P(k,l) = \{\{1,2\},\{1,3\}\}$, $c_{13} = 10 \leq c_{23} = 11$
   - $\{k,l\} = \{2,4\}$, $P(k,l) = \{\{1,2\},\{1,3\},\{3,5\},\{4,5\}\}$, $c_{13} = 10 \leq c_{24} = 12$
   - $\{k,l\} = \{2,5\}$, $P(k,l) = \{\{1,2\},\{1,3\},\{3,5\}\}$, $c_{13} = 10 \leq c_{25} = 10$
   - $\{k,l\} = \{4,6\}$, $P(k,l) = \{\{4,5\},\{4,6\}\}$, $c_{45} = 5 \leq c_{46} = 6$

6.4 Kruskal's Algorithm

The Kruskal's algorithm is an efficient algorithm that builds upon the path optimality conditions. It first sorts all the edges in non-decreasing order of their costs and defines a set, L, that is the set of edges chosen to be part of a MST. Initially, the set L is empty. It then examines the edges in the sorted order one-by-one and checks whether adding each edge to L creates a cycle with the edges already in L. If it does not, the edge is added to L; otherwise, it is discarded. It terminates when L contains $n - 1$ edges. At termination, the edges in L constitute a MST T^*.

The correctness of Kruskal's algorithm follows from the fact that it discards each non-tree edge $\{k, l\}$ with respect to T^* at some stage because it would create a cycle with the edges already in L. However, observe that the cost of edge $\{k, l\}$ is greater than or equal to the cost of every edge in that cycle because the edges are examined in non-decreasing order of their costs. Therefore, the spanning tree T^* satisfies the path optimality conditions, so it is a MST.

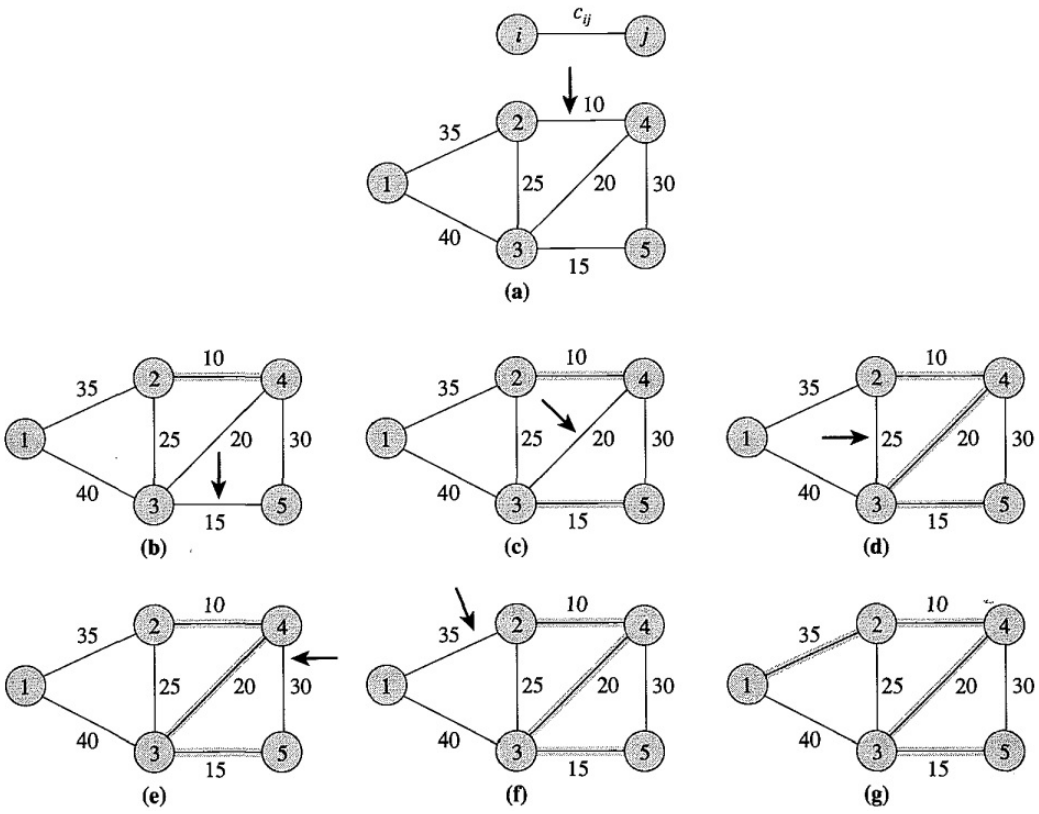
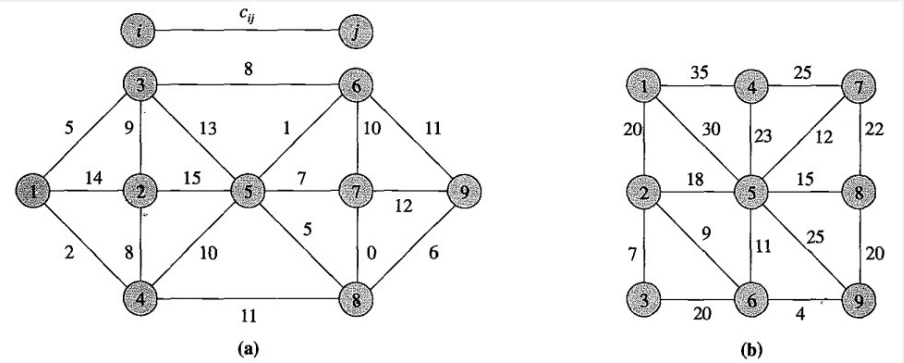


Figure 6.5: Illustrating Kruskal's algorithm

To illustrate Kruskal's algorithm on a numerical example, we consider the network shown in Figure 6.5(a). Sorted in the order of their costs, the edges are $\{2, 4\}$, $\{3, 5\}$, $\{3, 4\}$, $\{2, 3\}$, $\{4, 5\}$, $\{1, 2\}$, and $\{1, 3\}$. In the first three iterations, the algorithm adds the edges $\{2, 4\}$, $\{3, 5\}$, and $\{3, 4\}$ to L (see Figures 6.5(a)-(c)). In the next two iterations, the algorithm examines edges $\{2, 3\}$ and $\{4, 5\}$ and discards them because the addition of each edge to L creates a cycle (see Figure 6.5(d)-(e)). Then the algorithm adds edge $\{1, 2\}$ to L and terminates. Figure 6.5(g) shows the MST.

Exercise 6.2.

Using Kruskal's algorithm, find a MST of the following graphs



When Kruskal's algorithm is applied to the first graph, then the order in which the edges are included in the spanning tree is $\{7,8\}$, $\{5,6\}$, $\{1,4\}$, $\{1,3\}$, $\{5,8\}$, $\{8,9\}$, $\{3,6\}$, and $\{2,4\}$.

In the case of the second graph, the order in which edges are included in the spanning tree is $\{6,9\}$, $\{2,3\}$, $\{2,6\}$, $\{5,6\}$, $\{5,7\}$, $\{5,8\}$, $\{1,2\}$, and $\{4,5\}$.

6.5 Prim's Algorithm

Just as the path optimality conditions allowed us to develop Kruskal's algorithm, the cut optimality conditions permit us to develop another simple algorithm for the MSTP, known as Prim's algorithm. This algorithm builds a spanning tree from scratch by fanning out from a single node and adding edges one at a time. It maintains a tree spanning on a subset S of nodes and adds a nearest neighbor to S. The algorithm does so by identifying an edge $\{i,j\}$ of minimum cost in the cut $[S,\overline{S}]$. It adds edge $\{i,j\}$ to the tree, node j to S, and repeats this basic step until $S = N$.

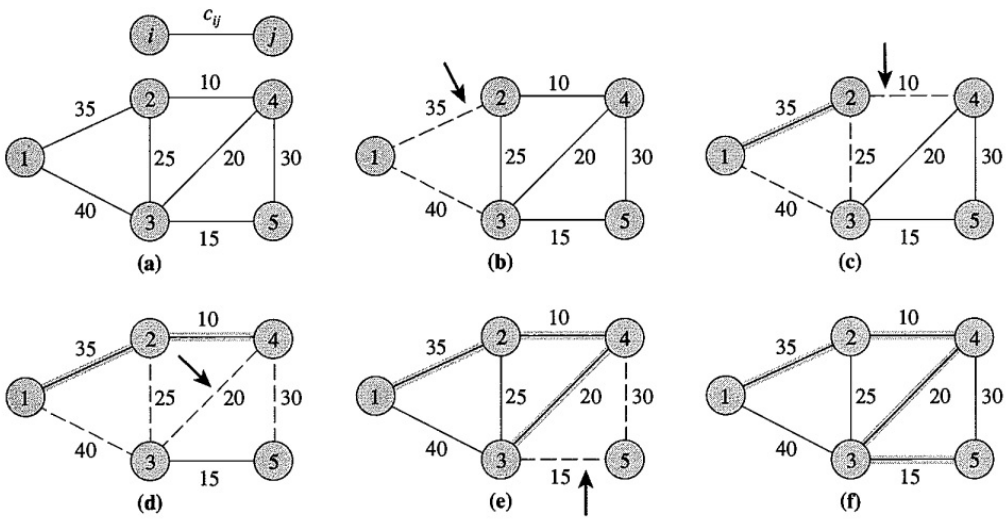
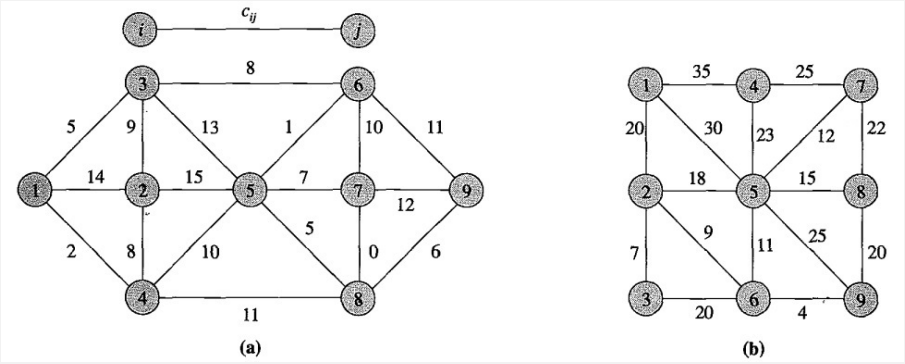


Figure 6.6: Illustrating Prim's algorithm

We illustrate Prim's algorithm on the same example, shown in Figure 6.6(a), that we used earlier to illustrate Kruskal's algorithm. Suppose, initially, that $S = \{1\}$. The cut $(S, \overline{S})$ contains two edges, $\{1, 2\}$ and $\{1, 3\}$, and the algorithm selects the edge $\{1, 2\}$ (see Figure 6.6(b)). At this point $S = \{1, 2\}$, and the cut $(S, \overline{S})$ contains the edges $\{1, 3\}$, $\{2, 3\}$, and $\{2, 4\}$. The algorithm selects edge $\{2, 4\}$ since it has the minimum cost among these three edges (see Figure 6.6(c)). In the next two iterations, the algorithm adds edge $\{3, 4\}$ and then edge $(3, 5)$; Figure 6.6(d)-(e) shows the details of these iterations. Figure 6.6(f) shows the MST produced by the algorithm.

Exercise 6.3.

Using Prim's algorithm, find a MST of the following graphs



When Prim's algorithm is applied to the first graph, by initially setting $S = \{1\}$, then the order in which the edges are included in the spanning tree is $\{1, 4\}$, $\{1, 3\}$, $\{2, 4\}$, $\{3, 6\}$, $\{5, 6\}$, $\{5, 8\}$, $\{7, 8\}$, and $\{8, 9\}$. In the case of the second graph, by initially setting $S = \{1\}$, the order in which edges are included in the spanning tree is $\{1, 2\}$, $\{2, 3\}$, $\{2, 6\}$, $\{6, 9\}$, $\{5, 6\}$, $\{5, 7\}$, $\{5, 8\}$, and $\{4, 5\}$.

6.6 Minimum Spanning Trees and Linear Programming

It is interesting to discuss how to formulate the MSTP as an ILP.

Let $E(S)$ denote the set of edges contained in the subgraph of $G = (N, E)$ induced by the node set S (i.e., $E(S)$ is the set of edges of E with both endpoints in S). Consider the following ILP formulation of the MSTP

$$\min \sum_{\{i,j\} \in E} c_{ij} x_{ij} \tag{6.2a}$$

$$\text{s.t.} \sum_{\{i,j\} \in E} x_{ij} = n - 1 \tag{6.2b}$$

$$\sum_{\{i,j\} \in E(S)} x_{ij} \leq |S| - 1 \quad \forall S \subset N : |S| \geq 3 \tag{6.2c}$$

$$x_{ij} \in \{0, 1\} \qquad \forall \{i, j\} \in E \tag{6.2d}$$

In this formulation, the binary variable x_{ij} indicates whether we select edge $\{i, j\}$ as part of the chosen spanning tree (note that constraints (6.2c) with $|S| = 2$ imply that each $x_{ij} \leq 1$). Constraint (6.2b) is a

cardinality constraint implying that we choose exactly $n - 1$ edges, and constraint (6.2c) imply that the set of chosen edges contain no cycles. As a function of the number of nodes in the network, this model contains an exponential number of constraints.

Exercise 6.4.

Suppose that you want to determine a spanning tree T that minimizes the objective function $\sqrt{\sum_{\{i,j\}\in T} c_{ij}^2}$. How would you solve this problem?

Observe that a spanning tree T that minimizes the objective function $\sqrt{\sum_{\{i,j\}\in T} c_{ij}^2}$ must also be optimal when the objective function is $\sum_{\{i,j\}\in T} c_{ij}^2$. To optimize this latter objective, one can identify a spanning tree in the graph G with the cost of each edge $\{i,j\}$ as c_{ij}^2.

Exercise 6.5.

Let T^* be a minimum spanning tree of a graph $G = (N, E)$.

Sensitivity Analysis For any edge $\{i,j\} \in E$, we define its cost interval as the set of values of c_{ij} for which T^* continues to be a minimum spanning tree. Describe a method for determining the cost interval of a given edge $\{i,j\} \in E$.

Arc Deletions How can you re-optimize the minimum spanning tree T^* after deleting an edge $\{i,j\} \in E$ from the graph?

Arc Additions How can you re-optimize the minimum spanning tree T^* after adding an edge $\{i,j\}$ to E?

Sensitivity Analysis There are two cases:

- When $\{i,j\} \in T^*$, then the maximum value of c_{ij} is the minimum value of the cost of any nontree edge in the cut formed upon removing $\{i,j\}$ from T^*. Let $\{k,\ell\}$ be such a nontree edge, then the cost interval of the edge $\{i,j\}$ is $[-\infty, c_{k\ell}]$;
- When $\{i,j\} \notin T^*$, then the minimum value of c_{ij} is equal to the maximum value of the cost of any tree edge in the cycle W formed by adding edge $\{i,j\}$ to T^*. Let $\{k,\ell\}$ be such a maximum cost tree edge in W, then the cost interval of edge $\{i,j\}$ is $[c_{k\ell}, +\infty]$.

Arc Deletions If the edge being deleted is a nontree edge, then the minimum spanning tree T^* does not change. On the other hand, if the edge $\{i,j\}$ being deleted is a tree edge, then we should obtain a minimum cost edge $\{k,\ell\}$ in the $i - j$ cut $(S, \overline{S})$ formed by removing edge $\{i,j\}$ from T^*. The new minimum spanning tree is simply the one formed by replacing edge $\{i,j\}$ in T^* by the edge $\{k,\ell\}$.

Arc Additions Let P be the unique path from node i to node j in T^* and edge $\{k,\ell\}$ be a maximum cost edge in P. If $c_{ij} \geq c_{k\ell}$, then T^* remains optimal; otherwise, replacing edge $\{k,\ell\}$ by the edge $\{i,j\}$ yields an optimal tree of the modified graph.

Exercise 6.6.

A spanning tree T is a *balanced spanning tree* if, among all spanning trees, the difference between the maximum edge cost in T and the minimum edge cost in T is as small as possible. Provide an ILP model of the problem.

$$\min \quad \beta - \alpha \tag{6.3a}$$

$$\text{s.t.} \sum_{\{i,j\} \in E} x_{ij} = n - 1 \tag{6.3b}$$

$$\sum_{\{i,j\} \in E(S)} x_{ij} \leq |S| - 1 \qquad \forall S \subset N \,:\, |S| \geq 3 \tag{6.3c}$$

$$c_{ij} x_{ij} \leq \beta \qquad \forall \{i,j\} \in E \tag{6.3d}$$

$$M - (M - c_{ij}) x_{ij} \geq \alpha \qquad \forall \{i,j\} \in E \tag{6.3e}$$

$$x_{ij} \in \{0,1\} \qquad \forall \{i,j\} \in E \tag{6.3f}$$

$$\alpha, \beta \in \mathbb{R}_+ \tag{6.3g}$$

where α and β are two auxiliary real variables representing the minimum and maximum cost of the edges of the selected spanning tree, and M is a sufficiently large number (e.g., $M = \max_{\{i,j\} \in E} \{c_{ij}\}$).