9
Finding the Optimum

9.1 Finding the Best Tree

A country with n towns wants to construct a new telephone network to connect all towns. Of course, they don't have to build a separate line between every pair of towns; but they do need to build a connected network; in our terms, this means that the graph of direct connections must form a connected graph. Let's assume that they don't want to build a direct line between towns that can be reached otherwise (there may be good reasons for doing so, as we shall see later, but at the moment let's assume their only goal is to get a connected network). Thus they want to build a minimal connected graph with these nodes, i.e., a tree.

We know that no matter which tree they choose to build, they have to construct $n - 1$ lines. Does this mean that it does not matter which tree they build? No, because lines are not equally easy to build. Some lines between towns may cost much more than some other lines, depending on how far the towns are, whether there are mountains or lakes between them, etc. So the task is to find a spanning tree whose total cost (the sum of costs of its edges) is minimal.

How do we know what these costs are? Well, this is not something mathematics can tell you; it is the job of the engineers and economists to estimate the cost of each possible line in advance. So we just assume that these costs are given.

At this point, the task seems trivial (very easy) again: Just compute the cost of each tree on these nodes, and select the tree with smallest cost.

We dispute the claim that this is easy. The number of trees to consider is enormous: We know by Cayley's Theorem (Theorem 8.3.1) that the number of labeled trees on n nodes is n^{n-2}. So for 10 cities, we'd have to look at 10^8 (one hundred million) possible trees; for 20 cities, the number is astronomical (more than 10^{20}). We have to find a better way to select an optimal tree; and that's the point where mathematics comes to the rescue.

There is this story about the pessimist and the optimist: They each get a box of assorted candies. The optimist always picks the best; the pessimist always eats the worst (to save the better candies for later). So the optimist always eats the best available candy, and the pessimist always eats the worst available candy; and yet, they end up with eating same candies.

So let's see how the optimistic government would build the telephone network. They start with raising money; as soon as they have enough money to build a line (the cheapest line), they build it. Then they wait until they have enough money to build a second connection. Then they wait until they have enough money to build a third connection... It may happen that the third-cheapest connection forms a triangle with the first two (say, three towns are close to each other). Then, of course, they skip this and raise enough money to build the fourth-cheapest connection.

At any time, the optimistic government will wait until they have enough money to build a connection between two towns that are not yet connected by any path, and build this connection.

Finally, they will get a connected graph on the n nodes representing the towns. The graph does not contain a cycle, since the edge of the cycle constructed last would connect two towns that are already accessible from each other through the other edges of the cycle. So, the graph they get is indeed a tree.

But is this network the least expensive possible? Could the cheap attitude at the beginning backfire and force the government to spend much more at the end? We'll prove below that our optimistic government has undeserved success: The tree they build is as inexpensive as possible.

Before we jump into the proof, we should discuss why we said that the government's success was "undeserved." We show that if we modify the task a little, the same optimistic approach might lead to very bad results.

Let us assume that for reasons of reliability, they require that between any two towns there should be at least two paths with no edge in common (this guarantees that when a line is inoperational because of failure or maintenance, any two towns can still be connected). For this, $n-1$ lines are not enough ($n-1$ edges forming a connected graph must form a tree, but then if any edge is deleted, the rest will not be connected any more). But n lines suffice: All we have to do is to draw a single cycle through all the towns. This leads to the following task:

Find a cycle with n given towns as nodes such that the total cost of constructing its edges is minimum.

(This problem is one of the most famous tasks in mathematical optimization: it is called the *Traveling Salesman Problem.* We'll say more about it later.)

Our optimistic government would do the following: Build the cheapest line, then the second cheapest, then the third cheapest, etc., skipping the construction of lines that are superfluous: It will not build a third edge out of a town that already has two, and will not build an edge completing a cycle unless this cycle connects all nodes. Eventually, they get a cycle through all towns, but this is not necessarily the best! Figure 9.1 shows an example where the optimistic method (called "greedy" in this area of applied mathematics) gives a cycle that is quite a bit worse than optimal.

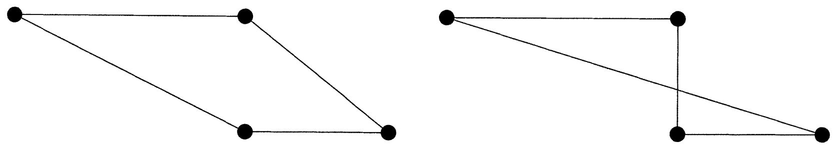
FIGURE 9.1. Failure of the greedy method. Construction costs are proportional to the distance. The first figure shows a cheapest (shortest) cycle through all 4 towns; the second shows the cycle obtained by the optimistic (greedy) method.

So the greedy method can be bad for the solution of a problem that is only slightly different from the problem of finding the cheapest tree. Thus the fact (to be proved below) that the optimistic government builds the best tree is indeed undeserved luck.

So let us return to the solution of the problem of finding a tree with minimum cost, and prove that the optimistic method yields a cheapest tree. The optimistic method is often called the *greedy Algorithm*; in the context of spanning trees, it is called *Kruskal's Algorithm.* Let us call the tree obtained by the greedy method the *greedy tree*, and denote it by F. In other words, we want to prove that any other tree would cost at least as much as the greedy tree (and so no one could accuse the government of wasting money and justify the accusation by exhibiting another tree that would have been cheaper).

So let G be any tree different from the greedy tree F. Let us imagine the process of constructing F, and the step when we first pick an edge that is not an edge of G. Let e be this edge. If we add e to G, we get a cycle C. This cycle is not fully contained in F, so it has an edge f that is not an edge of F (Figure 9.2). If we add the edge e to G and then delete f, we get a (third) tree H. (Why is H a tree? Give an argument!)

We want to show that H *is at most as expensive as* G. This clearly means that e is at most as expensive as f. Suppose (by indirect argument) that f is cheaper than e.

Now comes a crucial question: Why didn't the optimistic government select f instead of e at this point in time? The only reason could be that

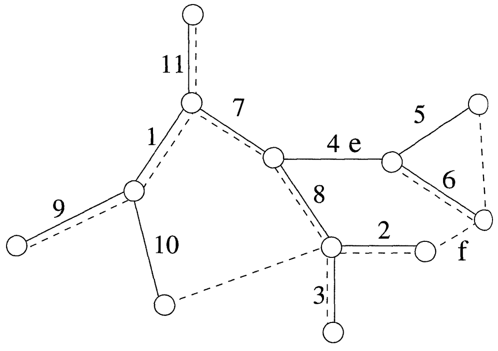

FIGURE 9.2. The greedy tree is optimal

f was ruled out because it would have formed a cycle C' with the edges of F already selected. But all these previously selected edges are edges of G, since we are inspecting the step when the first edge not in G was added to F. Since f itself is an edge of G, it follows that all edges of C' are edges of G, which is impossible, since G is a tree. This contradiction proves that f cannot be cheaper than e and hence G cannot be cheaper than H.

So we replace G by this tree H that is not more expensive. In addition, the new tree H has the advantage that it coincides with F in more edges, since we deleted from G an edge not in F and added an edge in F. This implies that if H is different from F and we repeat the same argument again and again, we get trees that are not more expensive than G, and coincide with F in more and more edges. Sooner of later we must end up with F itself, proving that F was no more expensive than G.

9.1.1 A pessimistic government could follow the following logic: If we are not careful, we may end up with having to build that extremely expensive connection through the mountain; so let us decide right away that building this connection is not an option, and mark it as "impossible." Similarly, let us find the second most expensive line and mark it "impossible," etc. Well, we cannot go on like this forever: We have to look at the graph formed by those edges that are still possible, and this "possibility graph" must stay connected. In other words, if deleting the most expensive edge that is still possible from the possibility graph would destroy the connectivity of this graph, then like it or not, we have to build this line. So we build this line (the pessimistic government ends up building the most expensive line among those that are still possible). Then they go on to find the most expensive line among those that are still possible and not yet built, mark it impossible if this does not disconnect the possibility graph, etc.

Prove that the pessimistic government will have the same total cost as the optimistic.

9.1.2 Formulate how the pessimistic government will construct a cycle through all towns. Show by an example that they don't always get the cheapest solution.

9.2 The Traveling Salesman Problem

Let us return to the question of finding a cheapest possible cycle through all the given towns: We have n towns (points) in the plane, and for any two of them we are given the "cost" of connecting them directly. We have to find a cycle with these nodes such that the cost of the cycle (the sum of the costs of its edges) is as small as possible.

This problem is one of the most important in the area of *combinatorial optimization*, the field dealing with finding the best possible design in various combinatorial situations, like finding the optimal tree discussed in the previous section. It is called the *Traveling Salesman Problem*, and it appears in many disguises. Its name comes from the version of the problem where a traveling salesman has to visit all towns in a region and then return to his home, and of course, he wants to minimize his travel costs. It is clear that mathematically, this is the same problem. It is easy to imagine that one and the same mathematical problem appears in connection with designing optimal delivery routes for mail, optimal routes for garbage collection, etc.

The following important question leads to the same mathematical problem, except on an entirely different scale. A machine has to drill a number of holes in a printed circuit board (this number could be in the thousands), and then return to the starting point. In this case, the important quantity is the *time* it takes to move the drilling head from one hole to the next, since the total time a given board has to spend on the machine determines the number of boards that can be processed in a day. So if we take the time needed to move the head from one hole to another as the "cost" of this edge, we need to find a cycle with the holes as nodes, and with minimum cost.

The Traveling Salesman Problem is closely related to Hamiltonian cycles. First of all, a traveling salesman tour is just a Hamiltonian cycle in the complete graph on the given set of nodes. But there is a more interesting connection: *The problem of whether a given graph has a Hamiltonian cycle can be reduced to the Traveling Salesman Problem.*

Let G be a graph with n nodes. We define the "distance" of two nodes as follows: adjacent nodes have distance 1; nonadjacent nodes have distance 2.

What do we know about the Traveling Salesman Problem on the set of nodes of G with this new distance function? If the graph contains a Hamiltonian cycle, then this is a traveling salesman tour of length n. If the graph has no Hamiltonian cycle, then the shortest traveling salesman tour has length at least $n + 1$. This shows that any algorithm that solves the Traveling Salesman Problem can be used to decide whether or not a given graph has a Hamiltonian cycle.

The Traveling Salesman Problem is much more difficult than the problem of finding the cheapest tree, and there is no algorithm to solve it that would be anywhere nearly as simple, elegant and efficient as the "optimistic" algorithm discussed in the previous section. There are methods that work quite well most of the time, but they are beyond the scope of this book.

But we want to show at least one simple algorithm that, even though it does not give the best solution, never loses more than a factor of 2. We describe this algorithm in the case where the cost of an edge is just its length, but it would not make any difference to consider any other measure (like time, or the price of a ticket), as long as the costs $c(ij)$ satisfy the *triangle inequality*:

$$c(ij) + c(jk) \geq c(ik). \tag{9.1}$$

(Distances in Euclidean geometry satisfy this condition by classical results in geometry: The shortest route between two points is a straight line. Airfares sometimes don't satisfy this inequality: It may be cheaper to fly from New York to Chicago to Philadelphia then to fly from New York to Philadelphia. But in this case, of course, we might consider the flight New York–Chicago–Philadelphia as one "edge," which does not count as a visit in Chicago. The distance function on a graph we introduced above when we discussed the connection between the Traveling Salesman Problem and Hamiltonian cycles satisfies the triangle inequality.)

We begin by solving a problem we know how to solve: Find a cheapest tree connecting up the given nodes. We can use any of the algorithms discussed in the previous section for this. So we find the cheapest tree T, with total cost c.

Now, how does this tree help in finding a tour? One thing we can do is to walk around the tree just as we did when constructing the "planar code" of a tree in the proof of Theorem 8.5.1 (see Figure 8.6). This certainly gives a walk that goes through each town at least once, and returns to the starting point.

Of course, this walk may pass through some of the towns more than once. But this is good for us: We can make shortcuts. If the walk takes us from i to j to k, and we have seen j already, we can proceed directly from i to k. Doing such shortcuts as long as we can, we end up with a tour that goes through every town exactly once (Figure 9.3). Let us call the algorithm described above the *Tree Shortcut Algorithm.*

Theorem 9.2.1 *If the costs in a Traveling Salesman Problem satisfy the triangle inequality (9.1), then the Tree Shortcut Algorithm finds a tour that costs less than twice as much as the optimum tour.*

Proof. The cost of the walk around the tree is exactly twice the cost c of T, since we used every edge twice. The triangle inequality guarantees that we have only shortened our walk by doing shortcuts, so the cost of the tour we found is not more than twice the cost of the cheapest spanning tree.

But we want to relate the cost of the tour we obtained to the cost of the optimum tour, not to the cost of the optimum spanning tree! Well, this is

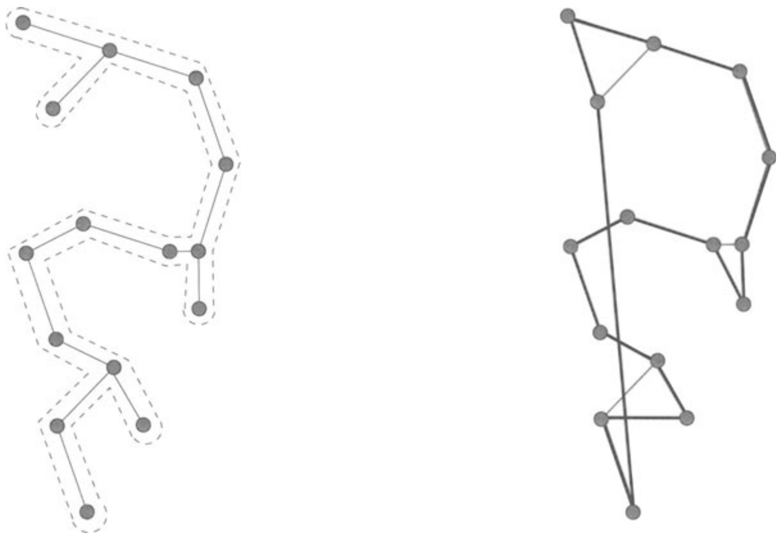

FIGURE 9.3. The cheapest tree connecting 15 given towns, the walk around it, and the tour produced by shortcuts (light edges on the right hand side are edges of the tree that are not used in the tour). Costs are proportional to distances.

easy now: *The cost of a cheapest spanning tree is always less than the cost of the cheapest tour.* Why? Because we can omit any edge of the cheapest tour to get a spanning tree. This is a very special kind of tree (a path), and as a spanning tree it may or may not be optimal. However, its cost is certainly not smaller than the cost of the cheapest tree, but smaller than the cost of the optimal tour, which proves the assertion above.

To sum up, the cost of the tour we constructed is at most twice that of the cheapest spanning tree, which in turn is less than twice the cost of a cheapest tour. □

9.2.1 Is the tour in Figure 9.3 shortest possible?

9.2.2 Prove that if all costs are proportional to distances, then a shortest tour cannot intersect itself.

Review Exercises

9.2.3 Prove that if all edge-costs are different, then there is only one cheapest tree.

9.2.4 Describe how you can find a spanning tree for which (a) the product of the edge-costs is minimal; (b) the maximum of the edge-costs is minimal.

9.2.5 In a real-life government, optimists and pessimists win in unpredictable order. This means that sometimes they build the cheapest line that does not create a cycle with those lines already constructed; sometimes they mark the most expensive lines "impossible" until they get to a line that cannot be marked impossible without disconnecting the network, and then they build it. Prove that they still end up with the same cost.

9.2.6 If the seat of the government is town r, then quite likely the first line constructed will be the cheapest line out of r (to some town s, say), then the cheapest line that connects either r or s to a new town, etc. In general, there will be a connected graph of telephone lines constructed on a subset S of the towns including the capital, and the next line will be the cheapest among all lines that connect S to a node outside S. Prove that the lucky government still obtains a cheapest possible tree.

9.2.7 Find the shortest tour through the points of a (a) 3×3 square grid; (b) 4×4 square grid; (c) 5×5 square grid; (d) generalize to $n \times m$ grids.

9.2.8 Show by an example that if we don't assume the triangle inequality, then the tour found by the Tree Shortcut Algorithm can be longer than 1000 times the optimum tour.