Chapter 1

THE GOLDEN TICKET

A CANDY MANUFACTURER DECIDES TO RUN A CONTEST and places a handful of golden tickets inside its chocolate bars, hidden among the tens of millions of bars produced each year. The finders of these tickets will get a rare factory tour.

How do you find those tickets? You could buy up as many of the chocolate bars as you can. You can try using a magnet, but gold is not magnetic. Or you could hire thousands of people and give each of them a small stack of chocolate to sift through. It sounds silly, but not for Veruca Salt, who really wanted a golden ticket to visit Willie Wonka's chocolate factory.

Veruca's father, a rich business man, decided to buy up all the chocolate bars that he could find. This wasn't enough—just because you have a large mound of chocolate bars doesn't make the ticket any easier to find. Mr. Salt also had a factory and that factory had workers, and he wasn't afraid to use them to find that golden ticket hidden in the chocolate bars. He recounted how he found the ticket to the press:

> I'm in the peanut business, you see, and I've got about a hundred women working for me over at my joint, shelling peanuts for roasting and salting. That's what they do all day long, those women they sit there shelling peanuts. So I says to them, 'Okay girls', I says, 'from now on, you can stop shelling peanuts and start shelling the wrappers off these crazy candy bars instead!' And they did. I had every worker in the place yanking the paper off those bars of chocolate full speed ahead from morning till night.

> "But three days went by, and we had no luck. Oh, it was terrible! My little Veruca got more and more upset each day, and every time I went home she would scream at me, '*Where's my Golden Ticket! I want my Golden Ticket!*' And she would lie for hours on the floor, kicking and yelling in the most disturbing way. Well, sir, I just hated to see my little girl feeling unhappy like that, so I vowed I would keep up the search until I'd got her what she wanted. Then suddenly . . . on the evening of the fourth day, one of my women workers yelled, 'I've got it! A Golden Ticket!' And I said, 'Give it to me, quick!' and she did, and I rushed it home and gave it to my darling Veruca, and now she's all smiles, and we have a happy home once again."

Like Mr. Salt, no matter how you try to find that ticket, you will need a considerable amount of time, money, or luck. Maybe someday someone will develop a cheap device that will let you find that ticket quickly, or maybe such a device will never exist.

Now, ten million is a very small number to today's computers. If you digitize the candy bars into a database, a typical desktop computer could search this database in a fraction of a second. Computers work much faster than a human searching through candy, but the problems are also much bigger.

What's the largest digital collection of data we have? How about the entire Internet? When you include all the video, audio, emails, and everything else, you have about 1,000,000,000,000,000,000 bytes of information, give or take a couple of zeros. A byte is about one character typed from your keyboard. That number sounds very large, but remember, computers are also very fast. A typical laptop can process about a trillion operations a second, which means you could theoretically search the entire Internet in just under four months if you could store the entire Internet in your laptop's memory. Google, with its hundreds of thousands of fast computers, scours the Internet constantly.

If computers can search the entire Internet quickly, then it sounds like we've solved the digital version of finding the golden ticket. But often we need computers not just to search the data we already have but to search for possible solutions to problems.

Consider the plight of Mary, a traveling salesman working for the US Gavel Corporation in Washington, D.C. Starting in her home city, she needs to travel to the capitals of all the lower forty-eight states to get the

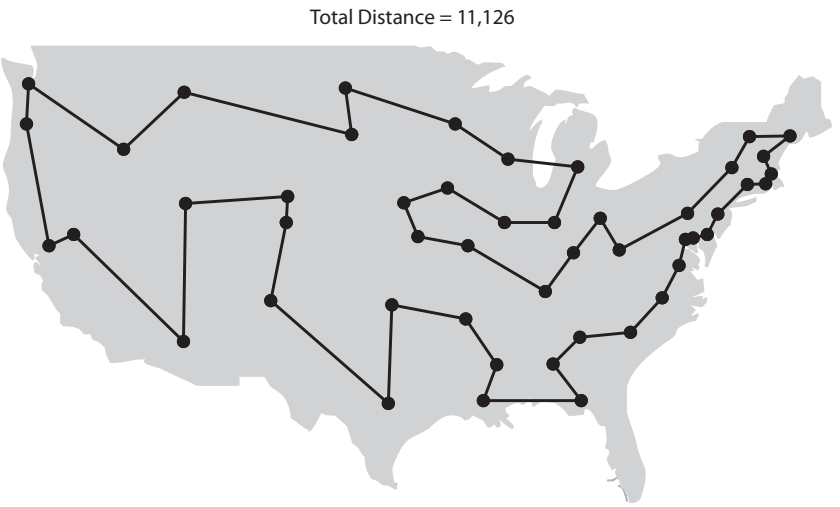

Figure 1-1. Traveling Salesman Problem Map.

state legislatures to buy an original US Gavel. US Gavel needs to reduce its travel expenses and asked Mary to find the best route through all the capitals that requires the smallest distance possible. Mary sketched a simple drawing on a map, played with it for a while, and came up with a pretty good route.

The travel department wanted Mary to see if she could come up with a different route, one that used less than 11,000 total miles. Mary wrote a computer program to try all possible different permutations of cities to find the quickest route, but a week later the program still hadn't finished. Mary sat down and did some calculations. There are forty-eight possible cities she could go to first, then she could choose among forty-seven of the remaining cities, then among the group of forty-six yet unvisited, and so on. That means a total of $48 \times 47 \times 46 \times \ldots \times 2 \times 1$ possible routes, equal to the sixty-two-digit number

12,413,915,592,536,072,670,862,289,047,373,375,038,521,486,354,677,
760,000,000,000.

If a computer could process one route in the time it takes light to cross the width of the smallest atom (about 0.00000000000000000033 seconds), it would still take about ten trillion trillion times the current

age of the universe to check all of them. No wonder Mary's laptop hadn't finished in a week. Mary wondered whether there was some better way to find the best route, to find that golden ticket among the candy bars of possible trips.

That's the basic question of this book. The P versus NP problem asks, among other things, whether we can quickly find the shortest route for a traveling salesman. P and NP are named after their technical definitions, but it's best not to think of them as mathematical objects but as concepts. "NP" is the collection of problems that have a solution that we want to find. "P" consists of the problems to which we can find a solution quickly. "P = NP" means we can always quickly compute these solutions, like finding the shortest route for a traveling salesman. "P ≠ NP" means we can't.

The Partition Puzzle

Consider the following thirty-eight numbers:

14,175, 15,055, 16,616, 17,495, 18,072, 19,390, 19,731, 22,161, 23,320, 23,717, 26,343, 28,725, 29,127, 32,257, 40,020, 41,867, 43,155, 46,298, 56,734, 57,176, 58,306, 61,848, 65,825, 66,042, 68,634, 69,189, 72,936, 74,287, 74,537, 81,942, 82,027, 82,623, 82,802, 82,988, 90,467, 97,042, 97,507, 99,564.

These thirty-eight numbers sum to 2,000,000. Can you break them into two groups of nineteen numbers, where each group of numbers sums to 1,000,000? Feel free to use your calculator or spreadsheet, or write a computer program. (The solution is given at the end of the chapter.)

Not so easy, is it. There are over 17 billion different ways to break these numbers into two groups. With some careful coding on today's fast computers you can find a solution. But suppose I gave you 3,800 different numbers or 38 million. No way will a simple computer program give you the answer now!

Is this just a silly math puzzle? What if there was some clever computer procedure that would let us quickly figure out how to break a list of numbers into two groups with the same sum if those groups exist? If so, we could use this procedure to do much more than solve these math

puzzles. We could use it to solve everything, even to find a traveling salesman's shortest route. This simple puzzle captures the P versus NP problem: any program that can solve large versions of this puzzle can compute just about anything.

The Hand

Your hands are the most incredible engineering devices every created. Your hands can poke, grab, and point. Your hands can tie shoes or shoot an arrow. Hands can play a piano or violin, perform tricks of magic, precisely steer a car, boat, train, or plane. Your hands can shake someone else's hands or thumb wrestle with them. Hands can speak through sign language or by the words they write or type. Hands can softly caress or violently attack. They can use the delicate tools of a watchmaker or manipulate a chain saw. The right hands can create great works of art, or music, or poetry. Nearly everything humans have achieved, they've achieved through their hands.

The hand has twenty-seven bones; five fingers, including the all-important thumb; and a complex arrangement of nerves, tendons, and muscles, all covered with a flexible skin. But this incredible device, a marvel of nature's engineering, can do nothing on its own. The hand performs only according to instructions it gets from the brain. A dead man's hands tell no tales, or do much of anything else.

The hand is nature's hardware, and hardware on its own cannot do much. The hand needs software, messages from the brain that tell the hands how to perform and accomplish the task the brain wishes the hands to do.

Yoky Matsuoka, a robotics professor at the University of Washington, runs a group that developed an anatomically correct robotic hand. The fingers have the full range of motions and movements that human fingers do. Physically, her robotic hand can perform any of the incredible things our hands can do, but in reality the robotic hand doesn't do more than very simple tasks. Writing computer programs to control all the various aspects of Matsuoka's hand is a complex task. Coordinating different muscles means even the simplest tasks require complex code.

Somehow our brain manages to control our hands. The brain is mostly a high-powered computer. If our brains can do it, there must also be computer programs that tell these hands how to tie shoes and execute great art.

Knowing that such programs exist and finding them are two different issues. In time, computer scientists will develop more and more sophisticated programs, and Matsuoka's hands will perform more complex activities. Perhaps someday those hands may perform tasks beyond what human hands can do. It will be an exciting journey, but likely a very slow one.

Does it have to be? Suppose we could simply describe a task and immediately have a program that provided that functionality. Feed in a movie of a human tying a knot, and immediately have the computer repeat the process with robotic hands. Give a computer the complete works of Shakespeare, and have that computer create a new "Shakespeare" play. Suppose whatever we can recognize we can find. We can if P = NP.

That's the excitement of the P versus NP problem. Can everything be made easy, or do we need to do some things the hard way? We can't rule it out. Nevertheless, we don't expect life to be so easy. We don't think that P = NP, but the possibility of such a beautiful world is tantalizing.

P versus NP

P versus NP is about all the problems described above and thousands more of a similar flavor: How fast can we search through a huge number of possibilities? How easily can we find that "golden ticket," that one best answer?

The P versus NP problem was first mentioned in a 1956 letter from Kurt Gödel to John von Neumann, two of the greatest mathematical minds of the twentieth century. That letter was unfortunately lost until the 1980s. The P versus NP problem was first publicly announced in the early 1970s by Steve Cook and Leonid Levin, working separately in countries on opposite sides of the Cold War. Richard Karp followed up with a list of twenty-one important problems that capture P versus NP, including the traveling salesman problem and the partition puzzle

mentioned earlier. After Karp, computer scientists started to realize the incredible importance of the P versus NP problem, and it dramatically changed the direction of computer science research. Today, P versus NP has become a critical question not just in computer science but in many other fields, including biology, medicine, economics, and physics.

The P versus NP problem has achieved the status of one of the great open problems in all of mathematics. Following the excitement of Andrew Wiles's 1994 proof of Fermat's Last Theorem, the Clay Mathematics Institute decided to run a contest for solutions to the most important unsolved mathematical problems. In 2000, the Clay Institute listed seven Millennium Problems and offered a $1 million bounty for each of them.

1. Birch and Swinnerton-Dyer conjecture
2. Hodge conjecture
3. Navier-Stokes equations
4. P versus NP
5. Poincaré conjecture
6. Riemann hypothesis
7. Yang-Mills theory

One of the Millennium Problems, the Poincaré conjecture, was solved by Grigori Perelman in 2003, though he has declined the $1 million prize. The other six, as of this writing, are unsolved.

Solve the P versus NP problem and get $1 million, a truly golden ticket!

Even better if you can show P = NP, for you then have a procedure that finds golden tickets, such as the solutions to all the other Millennium Prize Problems. Prove P = NP and get $6 million for solving the six unsolved Millennium Problems. Showing either P = NP or P ≠ NP won't be easy. If you want $6 million, you'll have a better chance playing the lottery.

Finding the Ticket

Sometimes we can find that golden ticket. Suppose I get in my car in Chicago and want to head to New York City. I plug the address into my GPS, which in a minute or two will find the fastest route from Chicago

to New York, and off I go. The full street map of the United States can fit in a few million bytes of memory, but the number of possible routes described by this map is much higher. How many routes are there between Chicago and New York? Even if we require the car to not go in the wrong direction, a conservative back-of-the-envelope calculation gives more than a vigintillion, or 1 followed by sixty-three zeros, different possible routes. Yet the GPS still finds the quickest route even though it doesn't have time to look through all these possibilities.

Travel times have an interesting property. Pick an intermediate location, say, Pittsburgh. The shortest trip from Chicago to New York through Pittsburgh is just the shortest trip from Chicago to Pittsburgh followed by the shortest trip from Pittsburgh to New York. There are faster routes that avoid Pittsburgh, but the shortest trip from Chicago to New York can't do worse than the shortest trip from Chicago to Pittsburgh and Pittsburgh to New York.

The GPS uses computer programs based on these properties to quickly narrow down the best route. The GPS may still need to examine tens or hundreds of millions of routes but nothing a computer processor can't handle, as opposed to the vigintillion number of total possibilities.

Finding the shortest path doesn't capture the full power of P versus NP. The shortest path problem tells us that just because there are a huge number of possibilities, we don't always need to explore them all. P versus NP asks us if we ever need to explore all the possibilities of any search problem.

The Long Road

In this book we tell the story of P and NP. What are P and NP? What kinds of problems do they capture? What are NP-complete problems, the hardest of all search problems? How do these problems capture the P versus NP question?

One simple example: What is the largest clique on Facebook, that is, the largest group of people all of whom are friends with each other? Is it a hundred people? A thousand? Even if you had access to all of

Facebook's data, this could be a difficult problem to solve. Finding large cliques is as hard as every search problem.

What happens if P = NP? We get a beautiful world where everything is easy to compute. We can quickly learn just about everything, and the great mysteries of the world fall quickly, from cures to deadly diseases to the nature of the universe. The beautiful world also has a dark underbelly, including the loss of privacy and jobs, as there is very little computers cannot figure out or accomplish.

The beautiful world is an unlikely outcome. That leaves us with hard search problems we still want or need to solve. We don't always need to give up. Computer scientists have developed many techniques, from heuristics that solve many of the problems most of the time to approximation techniques that give us close to ideal solutions.

How did we get to P and NP? This great story is actually two separate stories, taking place at a time when the world was divided by the Cold War. The ideas and questions of efficient computation were developed separately in those worlds, with two tales that lead to the same place, the P versus NP question.

How do we approach a proof that P ≠ NP? Kurt Gödel showed that mathematics cannot solve every problem. Can similar techniques show that there are search problems we can't solve quickly? Perhaps we can break computation down into its simplest components to analyze the difficulty of solving problems. Algebraic geometry, an abstract branch of mathematics, gives us some new hope, but we remain very far from settling this important question.

What good can come out of P ≠ NP? It can help us keep secrets and make fake random numbers that look really random.

Can future computers based on quantum mechanics make the P versus NP problem irrelevant? Not likely, but if built, they could solve some problems, like factoring large numbers, that remain out of reach for current machines. Quantum mechanics can also give us unbreakable secrets whether or not P = NP.

What about the future? The great challenges of computing still lie ahead of us. How do we deal with computers that have to work together to solve problems? How do we analyze the massive amount of data we generate every day? What will the world look like when we network

everything? The P versus NP problem only grows in importance as we try to tackle these challenges ahead.

Partition Puzzle Solution

The thirty-eight numbers

14,175, 15,055, 16,616, 17,495, 18,072, 19,390, 19,731, 22,161, 23,320, 23,717, 26,343, 28,725, 29,127, 32,257, 40,020, 41,867, 43,155, 46,298, 56,734, 57,176, 58,306, 61,848, 65,825, 66,042, 68,634, 69,189, 72936, 74,287, 74,537, 81,942, 82,027, 82,623, 82,802, 82,988, 90,467, 97,042, 97,507, 99,564

can be split into the following two groups:

15,055, 16,616, 19,390, 22,161, 26,343, 40,020, 41,867, 43,155, 46,298, 57,176, 58,306, 65,825, 66,042, 69,189, 74,537, 81,942, 82,623, 82,988, 90,467

and

14,175, 17,495, 18,072, 19,731, 23,320, 23,717, 28,725, 29,127, 32,257, 56,734, 61,848, 68,634 72,936, 74,287, 82,027, 82,802, 97,042, 97,507, 99,564.

Each of these groups of numbers sums to 1,000,000.