

DISCRETE MATHEMATICS TOPICS IN THE SECONDARY SCHOOL CURRICULUM

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science

in

The Department of Mathematics

by

Aimee Beth Boyd

B.S.E. in Secondary Mathematics Education, Auburn University, June 1999

May 2002

Acknowledgments

This thesis would not have been possible without several contributions. First and foremost, I would like to thank Dr. Bogdan Oporowski for giving his time and patience to guide me through this project. I would also like to thank the others on my committee: Dr. Frank Neubrandner, for giving me direction, and Dr. James Oxley, for teaching me to “know your audience.”

This work was motivated by both my love of teaching and discrete mathematics, and also by the fact that I had my own troubles with the abstract concepts of continuous mathematics.

It is a pleasure to thank Louisiana State University for providing me with a pleasant working environment. I would also like to thank everyone who contributed to this thesis in some way.

This thesis is dedicated to my family and friends for their support and encouragement, but above all, to my parents for their patience and understanding.

Table of Contents

- Acknowledgments** **ii**
- Abstract** **iv**
- Introduction** **1**
 - 0.1 The Different Types of Discrete Mathematics 3
 - 0.2 The Difference Between Discrete Mathematics and Continuous Mathematics 4
 - 0.3 How Discrete Mathematics Can Help Students 4
- Chapter 1. The Traveling Salesman Problem** **8**
 - 1.1 The TSP 8
 - 1.2 Solving the TSP 10
 - 1.3 Finding the Optimal Solution 11
 - 1.4 Approximation Algorithms: Saving Time 15
 - 1.4.1 Christofides' Algorithm 17
 - 1.4.2 Dijkstra's Algorithm 21
- Chapter 2. RSA Encryption** **25**
 - 2.1 Public Key Cryptography: RSA Encryption 28
 - 2.2 The Mathematics Behind RSA 30
 - 2.2.1 The Chinese Remainder Theorem 31
 - 2.2.2 Fermat's Little Theorem 39
 - 2.2.3 The Encryption and Decryption Processes 41
 - 2.3 Digital Signatures 44
- Conclusion** **46**
- References** **48**
- Vita** **50**

Abstract

This thesis discusses two topics of discrete mathematics in a manner suitable for presentation at a secondary-school level. The introduction outlines the benefits of including discrete mathematics in the secondary-school curriculum. The two chapters which follow include detailed treatment of the two selected topics: the Traveling Salesman Problem and RSA encryption.

The discussion of the Traveling Salesman Problem consists of introduction to the problem through several real-life scenarios, followed by a discussion of various methods for solving the problem. We discuss exact and approximation algorithms together with their computational complexity and practical limitations.

The discussion of RSA encryption begins with an introduction to the necessary background in number theory, which includes the Chinese Remainder Theorem, Bezout's Equation, and Fermat's Little Theorem. Following this is a discussion of encryption and decryption techniques, which are illustrated through an example that is accessible to any student with a basic scientific calculator.

Introduction

“Discrete mathematics grew out of the mathematical sciences’ response to the need for a better understanding of the combinatorial bases of the mathematics used in the development of efficient computer algorithms, the creation of new approaches to operations research problems, and the study of the heuristics underlying the approaches to such problems. The existence of discrete mathematics as a separate area of study began in the late 1960s” [3, p. 3].

Many school teachers, especially those teaching on the secondary level, are unfamiliar with discrete mathematics despite its growth in popularity over the past few years. The purpose of this thesis is to introduce discrete mathematics and help educate others on topics that can be implemented in the curriculum. We will introduce two important and widely known topics in this field and discuss the mathematics behind each one.

The first topic we will discuss is the Traveling Salesman Problem. This problem stems from graph theory and combinatorics, two areas which are very closely related to each other. Graph theory, the “mathematics of dots and lines,” involves such concepts as trees, cycles, and graph models, all of which are used in solving the Traveling Salesman Problem. It also involves such topics as graph coloring and Euler’s Formula, the last of which is very useful for studying polyhedra. Graph theory gives “students excellent examples of the importance of precise definitions, counting arguments, inductive proofs, algorithms, and real-world applications” [7, p. 87]. It is one of the major tools for modeling real-life problems in a rigorous mathematical setting. Such modeling helps students to develop the abstract think-

ing needed to solve real-life problems, which graph theory is particularly suited for.

Combinatorics, the mathematics of counting, involves three different types of problems, that also occur in graph theory [3, p. 1].

- 1) Existence problems, which ask whether a solution exists;
- 2) Counting problems, which ask how many solutions there are; and
- 3) Optimization problems, which ask whether an optimal solution exists.

Other combinatorics topics which are also suitable for secondary students, but which are not treated in detail here, include: combinations; counting principles; and inclusion/exclusion, which is a “method for counting the number of distinct elements in a finite union of sets” [4, p. 69].

The second topic we will discuss is RSA encryption, a form of public key cryptography. RSA encryption is based on number theory, which involves integers and their properties such as divisibility; greatest common divisor; modular arithmetic; and the Fundamental Theorem of Arithmetic, which states that every positive integer can be written uniquely as the product of prime numbers. The Fundamental Theorem of Arithmetic and modular arithmetic are important concepts used in RSA encryption. “Factoring integers into their prime factors is important in cryptology. Division of an integer by a positive integer produces a quotient and a remainder. Working with these remainders leads to modular arithmetic, which is used throughout computer science” [14, p. 113]. As we will see during the discussion of RSA encryption, factoring numbers into prime numbers is an essential part of the RSA algorithm.

Before proceeding with the two main topics of this thesis, some different areas of discrete mathematics will be introduced, along with the benefits discrete mathematics can offer secondary students. We will also discuss the difference between

discrete mathematics and continuous mathematics, the mathematics with which most students are familiar.

0.1 The Different Types of Discrete Mathematics

There are many areas of discrete mathematics other than graph theory, combinatorics, and number theory. The two areas described below are difference equations and matrices.

Difference equations are equations dealing with recursion. A *recursion* is “the process of defining something in terms of itself in spiral rather than a circular fashion. For example, the amount of money in an interest-paying bank account this year depends on how much was in the bank account last year, which depends on the amount that was in the account the previous year, and so on, until the spiral stops at the initial deposit made at time zero” [4, p. 71]. Difference equations are also known as recurrence equations or recurrence relations. They “are important because they describe change, and change is an essential characteristic of the real world. The standard tools used to describe continuous change are calculus and differential equations” [4, p. 71]. Difference equations allow students to work with differential equations without having to first learn calculus.

Using matrices can be of great assistance in the teaching of mathematics. Matrices can be used for such things as storing data, representing graphs, representing transformations such as rotations in the plane, solving systems of linear equations, and modeling games. They can be used in various courses such as algebra and calculus. “More instruction on matrices is probably the quickest and easiest way to get more discrete mathematics into the schools” [4, p. 72].

0.2 The Difference Between Discrete Mathematics and Continuous Mathematics

Continuous mathematics is what most secondary-school students know as mathematics. It is the mathematics that underlies most of high-school algebra and calculus. Continuous mathematics deals with the uncountable set, such as the reals, whereas discrete mathematics deals with countable, or finite sets of numbers, such as the integers or rationals. Discrete mathematics gives approximations for the size of some measurements. Continuous mathematics establishes bounds for the number of steps in computing algorithms that are finite in nature [3, p. 1]. For example, working with permutations allows one to find the number of existing possibilities, whereas in calculus, upper and lower bounds for a problem are found so that one knows the solution exists somewhere between the two.

Students learn about graphs involving e^x in algebra, and what an epsilon is in calculus, but what are these things? Many students have difficulty grasping the abstract concepts that permeate continuous mathematics, especially in advanced mathematics and calculus courses. Discrete mathematics has topics that students can easily relate to. Students work a problem, find an answer, and see the real-life application of it. Determining the best route a school bus can take to pick up children is a problem students can visualize. Most students do not see what taking the derivative of $3x^2$ is useful for, even if they are told. They have a hard time understanding the application because the concept is so abstract.

0.3 How Discrete Mathematics Can Help Students

The National Council of Teachers of Mathematics (NCTM) recommends that discrete mathematics be implemented into the curriculum as early as the seventh

grade, because it benefits students and helps maintain their interest in mathematics [5, p. 362]. But beyond what the NCTM recommends, is discrete mathematics really advantageous to students, and if so, why? Discussed below are some of the reasons.

First, it keeps students interested in mathematics. It helps entice them to regularly attend and participate in class. When students are interested in the material, it is easier for them to learn and stay focused when presented with tough, complex problems. While solving problems in discrete mathematics can be complicated, the problems themselves can be easily understood. The students, therefore, are able to understand and work the problems, which gives them a much needed confidence [12, pp. 35-36]. Many students are lost from mathematics forever during high school. Discrete mathematics is a great way to help these students stay interested and involved in mathematics.

Second, discrete mathematics benefits students by allowing them to see the connections between the mathematics they are studying and the real world. “For example, we might be able to convince our students that calculus can be used to help civil engineers build better bridges, but the students still might not see how it really works. But in graph theory [and other areas of discrete mathematics], we can explain the applications, the students can see how they work, and they can actually see real problems” [7, p. 94].

Teachers need to help students dismiss the idea that there is nothing new left to discover in mathematics and help them to look beyond basic arithmetic computation. Discrete mathematics is the mathematical foundation of computer science and is “used extensively in business, industry, and government. For example, difference equations are an essential mathematical tool for high-technology engineering firms, and matrices are indispensable for computer graphics” [4, p. 75].

Another benefit of discrete mathematics is that it enriches the traditional curriculum. It places more emphasis on teaching students to think mathematically and less emphasis on certain computational skills [1, p. 83]. Discrete mathematics lends itself to group work more easily than does traditional mathematics. It is also helpful to teachers because it gives them a new way of teaching elements in the curriculum, which may make the traditional concepts easier to teach and learn. By using discrete mathematics to teach already existing elements in the curriculum, it can help to change the way students view mathematics altogether. Below are some of the ways that discrete mathematics complements topics already in the traditional secondary-school curriculum:

- Algebraic skills are needed and reinforced throughout discrete mathematics.
- In geometry, graph theory can be used to enrich the study of polygons and polyhedra.
- Difference equations give use to the fascinating new geometry of fractals [4, p. 75].

The NCTM does not provide teachers with detailed guidelines. They only provide a set of goals and topics to cover in high school for discrete mathematics. Therefore, it is up to each individual teacher, or the mathematics department within each school, to decide how these topics should be implemented. Unfortunately, many teachers are unfamiliar, and even uncomfortable, with discrete mathematics. Thus, it can be difficult for them to know how to incorporate these topics into the curriculum. Due to the fact that discrete mathematics can be used to teach traditional elements in the curriculum, these topics can be covered in different ways throughout the school year, without having to set aside extra time to cover them.

While the implementation of discrete mathematics into the curriculum is not discussed here in detail, many references cited in this thesis give numerous ideas on how to do so. Some also give sample lessons and projects for different skill levels.

Some excellent resources are:

- *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **36** (eds. J.G. Rosenstein, D.S. Franzblau, and F.S. Roberts), American Mathematical Society; 1997.
- *Discrete Mathematics Across the Curriculum, K-12* (eds. M.J. Kenney, and C.R. Hirsch), NCTM, Inc, Reston, Virginia; 1991.
- *Mathematics Teacher*, a monthly journal magazine.

Chapter 1. The Traveling Salesman Problem

We are going to study an important problem from discrete mathematics, called the Traveling Salesman Problem, also referred to as the TSP. This is a well known problem involving graph theory and combinatorics. While the TSP could be presented in such a way as to challenge bright students, it could also be presented to lower level/remedial high school students that may or may not have a strong background in algebra. Problems such as the TSP help to build students' problem-solving skills and algorithmic approaches without their being completely aware of it. Since this problem can easily be related to real life, students are easily interested in it. This is true even though the theory behind the TSP is not simple at all; it is a subject of an ongoing research by many prominent mathematicians.

1.1 The TSP

The Traveling Salesman Problem can be expressed in many different ways. The first one introduced below is the original version and which will be the focus of our exploration of the TSP. The examples following demonstrate other ways to present this problem.

Traveling Salesman Problem

A salesman is planning a business trip that takes him to certain cities in which he has customers and then brings him back home to the city in which he started. Between some of the pairs of cities he has to visit, there is direct air service; between others there is not. Can he plan the trip so that he (a) begins and ends in the same city while visiting every other city only once, and (b) pays the lowest price in airfare possible? The key to this is not just finding a solution, but an optimal solution, the one with the lowest airfare.

Automated Teller Machine Problem

A bank has many ATM machines. Each day, a courier goes from machine to machine to make collections, gather computer information, and service the machines. In what order should the machines be visited so that the courier's route is the shortest possible? This problem arises in practice at many banks. One of the earliest banks to use the TSP algorithm, in the early days of ATMs, was the Shawmutt Bank in Boston [13].

“The TSP naturally answers a subproblem in many transportation and logistics applications, [such as] the scheduling of service calls at cable firms, the delivery of meals to home-bound persons, the scheduling of stacker cranes in warehouses, the routing of trucks for parcel post pickup, and a host of others”[18].

There are many other variations of the TSP which branch outside the area of mathematics. For example, the TSP can be applied to problems in genetic engineering and biochemistry. These problems are far too complicated to present in detail in the average high-school class, but it is something that could be touched upon so as to introduce the students to new topics and areas of study. For the more advanced students, this is something they can research themselves. Some easy-reading books are *Introduction to Graph Theory* by D.B. West, and *Discrete Mathematics and Its Applications* by K.H. Rosen. These books provide explanations and examples that make the material easy to understand. The web site www.math.princeton.edu/TSP is also a good resource for information on the TSP. It gives details on different TSP instances and has more information on how the TSP is being used in areas outside of mathematics.

Many TSP instances being solved in areas like genetic engineering and biochemistry are very complex and require the extensive use of a computer. The role that

computers play in helping with the TSP will be examined later when computer algorithms are discussed.

1.2 Solving the TSP

Since the TSP problem and the ATM problem are just different versions of one main problem, we can use one explanation for both. In order to solve this problem mathematically, it will be set up as a mathematical model—more specifically, in this case, as a graph problem. A *graph*, G , consists of a set $V(G)$ of vertices, and a set $E(G)$ of edges. Two vertices are *adjacent* when joined by a common edge, and likewise two edges are adjacent when joined by a common vertex. An example of a *simple graph*, simple implying that the graph has no loops or multiple edges, can be seen in Figure 1.1. A *loop* is defined as an edge that begins and ends at the same vertex. *Multiple edges* exist when there is more than one edge between the same two vertices.



FIGURE 1.1. Two Examples of Simple Graphs

Each city of the TSP to be visited is represented by a vertex of the graph G . The total number of vertices, n , is the *order* of G . Any two vertices are connected by an edge, provided the cities have direct air service. Cycles are an important part of finding the solution to the TSP. A *cycle* traverses some, or all, of the vertices of a graph along the edges without repeating vertices, and begins and ends at the same vertex. What the salesman seeks is the lowest-cost cycle that allows him to start at one vertex, visit every other vertex exactly once, and finish at the originating

vertex; such a cycle is called a *Hamiltonian cycle*. Figure 1.2 has two examples of the same graph, one with a lower-cost Hamiltonian cycle than the other.

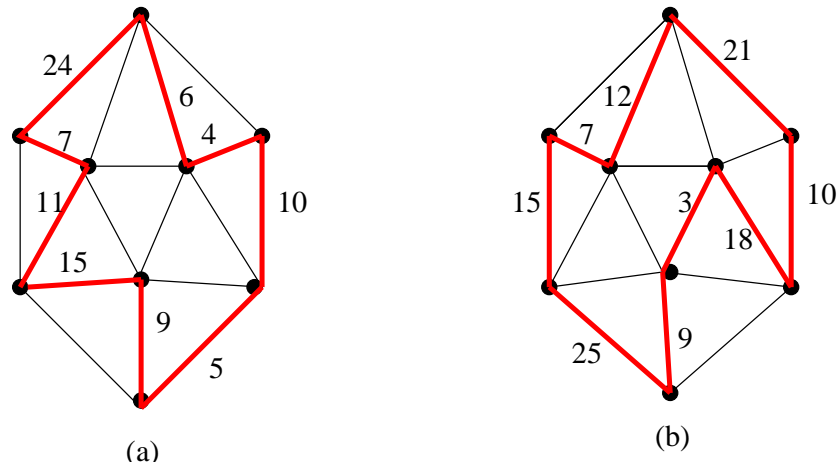


FIGURE 1.2. The Hamiltonian cycle found in graph (a) has a cost of 91, which is lower than the cost of the Hamiltonian cycle found in graph (b), which is 120.

The TSP problem that has been defined here deals with finding a Hamiltonian cycle with the lowest airfare. Any sort of weight, such as distance or cost, could be assigned to the edges. So while what is being looked for may change, the procedure for solving the problem does not. We are taking a real-life problem and solving it using a mathematical approach.

1.3 Finding the Optimal Solution

Suppose there is a problem with five cities for which an optimal solution is needed. It is possible to write out every arrangement of all five cities by hand, including the ones in which two cities do not have direct air service. Though this may be somewhat time-consuming, it can be done.

For example, suppose a salesman had to visit five cities: (A) Atlanta, (B) Boston, (C) Chicago, (D) Dallas, and (E) Eugene (Oregon). The graph for this problem is seen in Figure 1.3. It shows which cities have direct air service and the cost of each flight. First, we must pick a city to start in. An easy way of doing this is

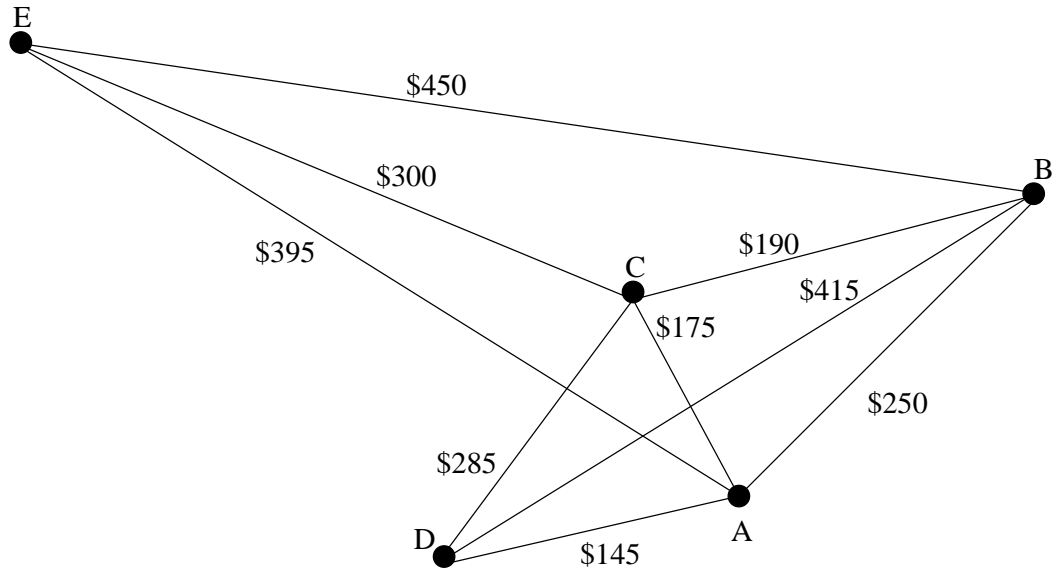


FIGURE 1.3. Five-city figure

by alphabetical order, so let (A) Atlanta be the first city. Now find every possible arrangement with Atlanta being first, paying no attention to whether two cities have direct air service or not.

A B C D E A	A D B C E A	A C B D E A	A E B C D A
A B C E D A	A D B E C A	A C B E D A	A E B D C A
A B D C E A	A D C B E A	A C D B E A	A E C B D A
A B D E C A	A D C E B A	A C D E B A	A E C D B A
A B E C D A	A D E B C A	A C E B D A	A E D B C A
A B E D E A	A D E C B A	A C E D B A	A E D C B A

Next, we must list all possible arrangements with B, C, D, and E as the starting point. This can easily be done following the example above. After finding every possible arrangement of all five cities, sort through each one until an optimal solution is found. The technique for doing so is explained below.

Since any city could be chosen to start in, there are five choices for the first city. Then, for the second city, there are four possibilities to choose from. For the third there are three to choose from, and so on. Multiplying these numbers together, $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$, gives the total number of possible arrangements that exist for this problem. It is more commonly written as $5!$, pronounced “five factorial”. So the number of arrangements is the total number of ways in which the different cities can be arranged so that each city is visited only once. In other words, the total number of arrangements is the number of *permutations* of the total number of cities. If it is known that the salesman will start in his hometown, the originating city is already chosen; thus, there is only one choice for the first city. But there still exist four choices for the second city, three for the third, and so on. So there are $4!$ different arrangements to check, which may or may not be possible solutions. To reduce this number even more, the cost of any arrangement of cities traveled one way is the same as the cost of its reversal; therefore, the two arrangements are only counted as one. For example, $ABCDEA$ and $AEDCBA$ are counted as one arrangement. Since every arrangement has a reversal, the number of remaining solutions is divided in half. Thus, the number of possible arrangements left to check is $\frac{4!}{2}$. Looking at all of the remaining possibilities, disregard the ones in which any two of the cities do not have direct air service. So only the arrangements with Hamiltonian cycles are left, and the optimal solution with the lowest airfare can be found.

Finding a solution in this manner is sometimes informally referred to as the *brute-force method*. For a number such as five, despite being tedious and somewhat time-consuming, using brute force is a viable method. But as the number of cities, n , grows, the number $\frac{(n-1)!}{2}$ also grows, and it grows extremely rapidly. For example, for 10 cities there would be, at most, 181,440 possible arrangements to check to

find an optimal solution, depending on how many arrangements can be disregarded due to any two cities not having direct air service. Therefore, this process is best used with a smaller number of cities. While using brute force may take some time, it will yield an optimal solution, which is something not all computer algorithms can give.

Using the brute-force method involves using heuristics. *Heuristics* are clever, logical ideas for working problems. When using them to help solve TSP instances, they are not guaranteed to give a solution and each heuristic may not work for every problem. When a problem is initially read, ideas are formulated on how to go about working the problem. For example, when looking at a TSP instance that involves covering the least amount of miles when driving across the country while stopping at certain cities, one heuristic is that one should travel from one coast to the other visiting the different cities along the way as opposed to traveling back and forth from coast to coast many times. Note that many heuristics are not as simple as the one just given, and many come only with time after working on several problems.

There does exist a computer code that may yield an optimal solution, though it is not guaranteed to do so. Its name, Concorde, stands for Combinatorial Optimization and Networked Combinatorial Optimization Research and Development Environment. It has given an optimal solution for a problem involving 15,112 cities throughout Germany. It took a total of 22.6 years to complete and was finally solved in 2001. Before this, an optimal solution for 13,509 cities throughout the United States was found in 1998. The 15,112 city tour is the largest recorded TSP instance to ever have been solved, and took a network of 110 processors at Princeton University and Rice University [18]. Thus, one can see that while an optimal solution was found, it took an enormous amount of time.

Concorde is a computer code involving the use of brute force and many different heuristics. The more difficult the problem, the more difficult the process of finding a solution. More about Concorde can be found on the following web site: www.math.princeton.edu/TSP. There one can find information concerning large instances of the TSP, and it is continually updated with new information. The computer code is available for downloading at the site, with certain restrictions.

Below, faster algorithms for finding Hamiltonian cycles are discussed, though they will not always yield an optimal solution.

1.4 Approximation Algorithms: Saving Time

For large numbers, such as 50, or even 25, using brute force is out of the question unless one has a great deal of free time. This is because every possible arrangement is being considered. “For example, with 25 vertices, a total of $\frac{24!}{2}$ (approximately 3.1×10^{23}) different Hamiltonian cycles would have to be considered. If it took just one nanosecond (10^{-9} second) to examine each Hamiltonian cycle, a total of approximately 10 million years would be required to find a [minimum-cost Hamiltonian cycle]” [14, p. 498]. So as the number of cities grows in the problems being solved, the method used changes. Despite all of the advances in technology, the time it takes for an algorithm used on a computer to find an optimal solution can still be considerably long.

Since no fast algorithm has been found that can guarantee an optimal Hamiltonian cycle in a reasonable amount of time, a fast algorithm that can yield something close to the desired solution is settled for. An algorithm that can do this is known as an *approximation algorithm*. More than one of these types of algorithms exists, and as was said above, they “do not necessarily produce the exact solution to the problem but instead are guaranteed to produce a solution which is close to an exact

solution” [14, p. 498]. Sometimes a close approximation to the optimal solution is all that is needed.

Since a common goal is to minimize the time it takes to find a solution, fast algorithms are more commonly known as *good* algorithms. The term *good* algorithm was coined by a man named Edmonds in order “to describe solution methods whose running time increases at most polynomially with problem size” [9, p. 10]. Algorithms can run in polynomial time and exponential time, but before these definitions can be given, *running time* must be defined.

Running time is expressed “as a function of the number of cities,” rather than being determined for each TSP instance. This means that running time is approximated for instances with different numbers of cities. So there is an approximated running time for a TSP instance with five cities, ten cities, etc. We will use O-notation (‘big O’ notation) which counts ‘steps’ instead of machine cycles. Therefore, the time is independent of the speed of the computer being used. One can think of one step as being one city in each arrangement of a TSP problem. For example, in the five-city problem above, each list, or arrangement, contained six cities. Therefore, every list consisted of six steps. To find the total number of steps the problem took, the total number of possible Hamiltonian cycles must be known. There is at most $\frac{4!}{2} = 12$ different lists that could be the optimal solution. So multiplying that number, 12, by the number of steps in each list, 6, yields the total number of steps that could have occurred in the five-city problem, which is 72. If the optimal solution is found to be the second or third list checked, then this problem would contain less than 72 steps.

If an algorithm’s running time is denoted by $O(f(n))$, then there is some constant c such that for any n , the running time is bounded [from above] by $cf(n)$ [9, p.

Function	Approximate Values		
n	10	100	1000
$n \log n$	10	200	3000
n^3	1000	1,000,000	10^9
$10^6 n^8$	10^{14}	10^{22}	10^{30}
2^n	1024	1.27×10^{30}	1.057×10^{301}
$n^{\log n}$	10	10,000	10,000,000,000
$n!$	3,628,800	9.33×10^{158}	4×10^{2567}

TABLE 1.1. A comparison of the growth of some polynomial functions (above) to that of some exponential functions (below) [9, p. 44].

42]. The speed of the computer is denoted by c , and n is the number of cities in the TSP instance.

An example of some polynomial-time and exponential-time functions can be viewed in Table 1.1, which compares their growth. Algorithms running in polynomial time are considered better than those that run in exponential time. This is because, for small values of n , polynomial and exponential functions may yield similar values, but as n grows large, the exponential function outputs a much greater number than the polynomial function. Thus, algorithms running in polynomial time have a steadier growth and are more useful for large numbers, which is why they are considered *good* algorithms.

1.4.1 Christofides' Algorithm

There exist approximation algorithms that will produce a Hamiltonian cycle with a total cost of W' with $W \leq W' \leq aW$, “where W is the total length of an exact solution and a is a constant” [14, p. 498]. In 1976 Christofides presented an algorithm which has a polynomial time complexity and which gives a solution that is guaranteed to be within the factor of $3/2$ of the optimal solution. This algorithm, in the worst case, gives a solution that is 50% larger than the optimal solution and “is still the best worst-case result known today” [9, p. 13]. Christofides' Algorithm is an improvement on the minimum spanning tree algorithm, which has

a polynomial time complexity and in the worst case yields a solution of twice the minimum spanning tree.

Before we describe these two algorithms, some terminology needs to be defined. A *trail* is formed by traversing along the edges of a graph where none of the edges are repeated. It does not have to contain every vertex of the graph, only the vertices in which it begins and ends, and every vertex between the two. A trail that begins and ends at the same vertex, a *closed trail*, and that contains every edge is called an *Euler tour*. Any graph with an Euler tour is an *Eulerian graph* [17]. These graphs have all vertices of even degree. The *degree* of a vertex is found by counting the number of edges which are adjacent to it. Thus, if a vertex has a degree of 2, then it has two edges connecting it to two different vertices.

A connected graph with no cycles is known as a *tree*. A *spanning tree* of a graph G is a tree which contains all of the vertices of G . So, while the spanning tree contains all of the original vertices of G , it may or may not contain all of the edges that are found in G . A *minimum spanning tree* is a spanning tree with the smallest-overall cost. Two examples of trees can be seen in Figure 1.4. While tree structures can vary greatly, the only thing they must all have in common is that they must be connected with no cycles.

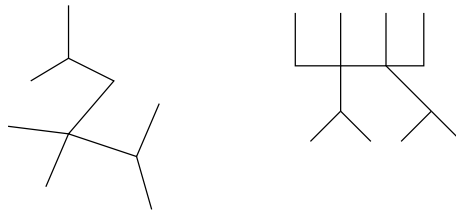


FIGURE 1.4. Trees

Knowledge of the *triangle-inequality* is necessary for the minimum spanning tree algorithm and Christofides' Algorithm. The triangle-inequality compares the dis-

tances of vertices on a graph. It states that the distance from vertex A to vertex C is less than or equal to the distance from A to another vertex B , added to the distance from B to C . More specifically, $d(A, C) \leq d(A, B) + d(B, C)$, where $d(A, C)$ represents the distance from vertex A to vertex C .

Since these algorithms involve the triangle-inequality, they are able to make use of *shortcuts*. A shortcut between two vertices may be taken instead of going through a third city. This is done for two reasons: one is that each city, other than the first and last, is only allowed to be visited once in order for a Hamiltonian cycle to exist; and the second is that using shortcuts saves distance, since the shortest distance between two points is a straight line. An example of this is seen below in Figure 1.5.

It must be noted that because Christofides' algorithm and the minimum spanning tree algorithm make use of the triangle-inequality, these algorithms may not be used for certain TSP instances. This is because the triangle-inequality is based on distance. The TSP instance that this thesis focuses on is based on the cost of airfare. In many cases it may cost less to have a connecting flight from city A to city B , then B to city C , than to fly directly from A to C . So depending on the weights used in the TSP instance, these algorithms may or may not be used.

Now we are ready for the minimum spanning tree algorithm.

Minimum Spanning Tree Algorithm:

Start with a minimum spanning tree, double its edges in order to obtain an Eulerian graph, find an Euler tour for this graph, and then convert it to a traveling salesman tour by using shortcuts. By the triangle-inequality the traveling salesman tour can be no longer than the Euler tour—hence, at most, twice the length of a minimum spanning tree [9, p. 160].

Christofides improved on this algorithm by using a better method for finding an Eulerian graph. He used the concept of a *minimum weight matching*. “Given a set containing an even number of cities, a *matching* is a collection of edges M such that each city is the endpoint of exactly one edge in M . A *minimum weight matching* is one for which the total length of the edges is minimum. Such matchings can be found in time $O(n^3)$ ” [9, p. 160]. The steps of Christofides’ algorithm are shown below. In Figure 1.5 the algorithm can be seen in action.

Christofides’ Algorithm:

Step 1: Construct a minimum spanning tree T on the set of all cities in I . The running time for this is $O(n^2)$. (Here I denotes the given TSP instance.)

Step 2: Construct a minimum matching M^* for the set of all odd-degree vertices in T . Here the running time is $O(n^3)$.

Step 3: (a) Find an Euler tour for the Eulerian graph that is the union of T and M^* , and (b) convert it to a tour using shortcuts. The running time here is $O(n^2)$.

“The running time for the algorithm is dominated by the time for finding the matching in Step 2, and hence is $O(n^3)$. The above arguments yield the following theorem about the algorithm’s worst-case behavior, where $C(I)$ stands for the length of the tour constructed when Christofides’ algorithm is applied to instance I .” The length of the Eulerian graph constructed is $\frac{3}{2}OPT(I)$, where OPT stands for optimum value [9, p. 162].

Theorem 1.1. *For any instance I of the TSP which obeys the triangle-inequality,*

$$C(I) \leq \frac{3}{2}OPT(I).$$

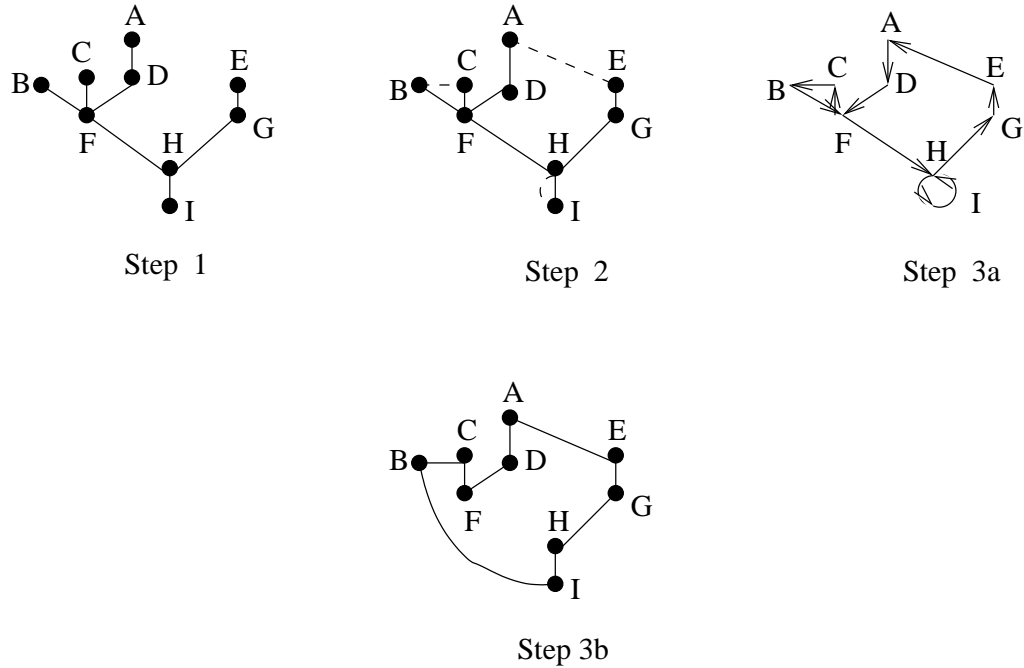


FIGURE 1.5. Snapshots of Christofides' Algorithm in action [9, p. 161]. (a) Minimum spanning tree T . (b) T plus a minimum cost matching of its odd-degree vertices. (c) Euler tour of (b): IHGEADFCBFHL. (d) Tour constructed by Christofides' algorithm, using shortcuts: IHGEADFCBI.

1.4.2 Dijkstra's Algorithm

Another algorithm used to solve the TSP was developed in 1959 by Dutch mathematician E. Dijkstra. His algorithm is a shortest path algorithm, meaning that it finds the length of the shortest path from a to a first vertex, the length of a shortest path a to a second vertex, and so on until the length of the shortest path from a to z is found. In this example, Dijkstra's Algorithm will be used to find the path having the lowest cost from a to z . This algorithm relies on a series of iterations in which a "distinguished set of vertices is constructed by adding one vertex at each iteration. A labeling procedure is carried out at each iteration. In this labeling procedure, a vertex w is labeled with the lowest cost of the path from a to w that contains only vertices already in the distinguished set. The vertex added to the distinguished set is one with a minimal label among those vertices

not already in the set” [14, p. 493]. This can be seen more clearly in the following example.

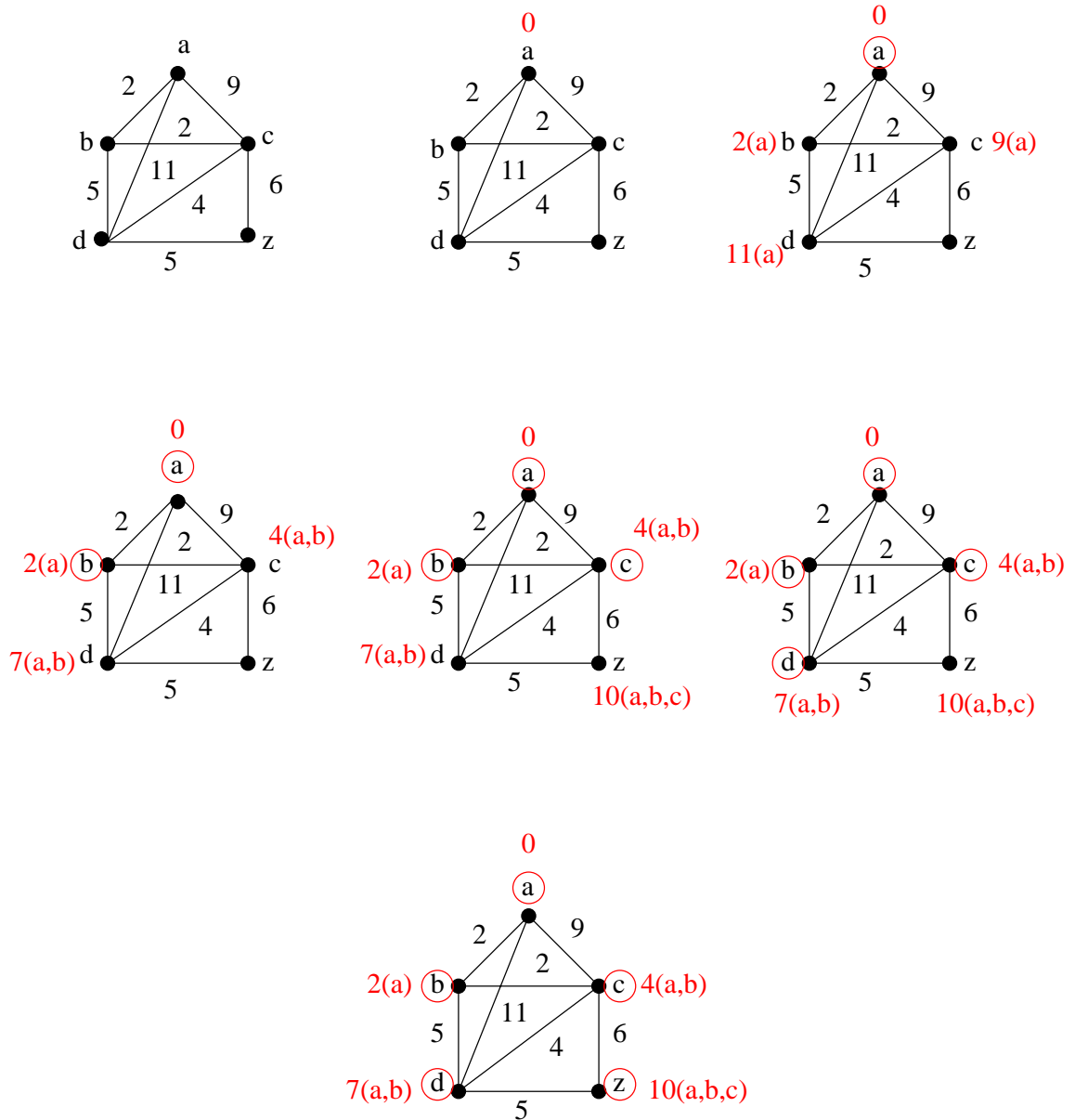


FIGURE 1.6. Dijkstra's Algorithm

For our example, Dijkstra's Algorithm will be used to find the path of lowest cost from vertex a to vertex z in the weighted graph shown in Figure 1.6. While the graph is small enough so that the path of lowest-cost could be found simply

by examining it, it was chosen this size for the purpose of the example. The figure shows the steps used by Dijkstra's Algorithm to find the lowest cost between a and z . At each iteration, the vertices of the distinguished set S_k , which is the set of vertices that have been chosen, are circled. The lowest cost path from a to each vertex containing only vertices from S_k (but not necessarily all of the vertices) is indicated for each iteration. The algorithm then terminates when z is circled. We will show that the path with the lowest cost from a to z is a, b, c, z with a cost of 10.

First, vertex a is labeled with 0. This is because at the 0th iteration the cost of the path from a to itself is 0, and so $S_0 = a$.

Next look at the costs of all the paths from a to another vertex via a direct path. There is the path from a to b with a cost of 2, a to c with a cost of 9, and a to d with a cost of 11. Since the lowest cost is 2 along the path a, b , vertex b is now circled and labeled with $2(a)$, and the distinguished set of vertices available for the next iteration is $S_1 = a, b$.

Continuing on, we will find the path of lowest cost from a to another vertex. Any path can be chosen containing one or more of the vertices from the distinguished set S_1 and extending to a new vertex. One of the following two paths will be chosen: a, b, c with a cost of 4, or a, d , with a cost of 11. Since the lowest cost is 4 along the path of a, b, c , vertex c is circled and labeled with $4(a, b)$.

For the third iteration, a path starting at a , containing any of the following vertices, $S_2 = a, b, c$, and then extending to a new vertex will be chosen. There is the path a, b, d with a cost of 7, or the path a, b, c, z with a cost of 10. The path with a cost of 7 is chosen, so vertex d is circled and then labeled with $7(a, b)$.

For the fourth iteration there is only one more vertex to add, which is z . After examining the costs of all the different paths containing the vertices within the set

$S_3 = a, b, c, d$ that extends to z , one can see that the path with the lowest cost is a, b, c, z with a cost of 10. Thus, in the final step, vertex z is circled and labeled with $10(a, b, c)$.

Hence it has been shown that the path from a to z with the lowest cost in this weighted graph is a, b, c, z of a cost of 10.

Chapter 2. RSA Encryption

Cryptography is a topic that will challenge bright students and take away the feeling that what they are studying is mundane and boring. This is also a topic that can be simplified for lower-level students. Most children play some kind of game that uses secret codes, and almost everyone has a computer. The challenging part of cryptography is creating a key that would prevent an unauthorized decryption of the message. This is another example of mathematics that could interest many students.

The topic being covered in this chapter is centered around public key cryptography. This allows people to exchange information while keeping it a secret from other sources. Only persons having the decryption key are able to decode the message. The main form of public key cryptography that is of our interest is called RSA encryption. This allows two people to exchange secret messages without ever having to exchange encryption and decryption keys. It is the most widely known form of public key cryptography. More on this and RSA encryption will be covered later. First, we will discuss more about cryptography in general. Below are two scenarios that use some form of public key cryptography.

Scenario 1:

An ambassador has to send information to his government concerning the foreign country he is in. The information has to be encrypted so that if the foreign government were to intercept the information, they would not be able to decrypt the transmission, or at least have trouble in doing so. How can this be done?

Scenario 2:

A man goes to Priceline.com to buy plane tickets. When he purchases the tickets, he has to buy them online using his credit card. If he does this, how can he be sure that when entering his credit card number, no one other than Priceline.com will have access to it?

Anyone who has ever entered a password on a computer or bought something online has used a form of cryptography. “Cryptography has proven to be the best technique for protecting information transmitted through accessible communication networks such as telephone lines, satellites,” and computers [2]. While cryptography once stayed within the realms of the government, mainly for military purposes, it is now reaching out into the world of mathematics. This is because mathematics “provides the theoretical justification behind the strength of a particular encryption system” [2]. This is a topic that has very detailed mathematical operations which may not be immediately obvious.

Here, a small example of cryptography will be performed using congruences. The words in the message will be translated into numbers, then those numbers will be translated into other numbers by using, in this case, a shift transformation of modulus 26. This is also referred to as an alphabetic shift cipher.

The encryption key, k , can be any number, and one letter of the word after translation into a number is p . The following equation is used to obtain the encrypted letter:

$$c = (p + k) \pmod{26}.$$

Decryption is done by using this equation:

$$p = (c - k) \pmod{26}.$$

Let MATH be the message to be sent. Since M is the 14th letter of the alphabet, $M = 14$. Now do the same for the following letters: $A = 01$, $T = 19$, and $H = 08$. So as not to make this example too complicated, a small key, $k = 14$, is chosen.

Now encrypt each letter, starting with M and continuing on with the rest.

$$c = (14 + 14) \pmod{26} = 28 \pmod{26} = 02.$$

$$c = (01 + 14) \pmod{26} = 15 \pmod{26} = 15.$$

$$c = (19 + 14) \pmod{26} = 33 \pmod{26} = 07.$$

$$c = (08 + 14) \pmod{26} = 22 \pmod{26} = 22.$$

The encrypted message is 02150722.

Suppose this message is sent to Susan. In order for her to decrypt the message using the formula for decryption, she would need to know what the encryption key, k , is. Then she would break up the encrypted message into blocks of two and put each block into the decryption formula as c in order to find its corresponding letter. This is seen by the following:

$$p = (02 - 14) \pmod{26} = -12 \pmod{26} = 14, \text{ which corresponds to the letter } M.$$

$$p = (15 - 14) \pmod{26} = 01 \pmod{26} = 01, \text{ which corresponds to the letter } A.$$

$$p = (07 - 14) \pmod{26} = -7 \pmod{26} = 19, \text{ which corresponds to the letter } T.$$

$$p = (22 - 14) \pmod{26} = 08 \pmod{26} = 08, \text{ which corresponds to the letter } H.$$

Thus, after decryption, Susan will have the original message which was sent to her.

This is a rather simple form of cryptography and is obviously not one that is very secure, due to the fact that the encryption and decryption keys are the same. Any cipher having the same encryption and decryption keys is known as a symmetric cipher. With the alphabetical shift cipher, it is common knowledge what each letter is shifted to: $A = 01$, $B = 02$, ..., $Z = 26$. Of course the shift can always be used backwards with $Z = 01$, and so on, and it is also possible to assign any number

or letter to each letter of the message to make decryption of the message more difficult.

Another drawback to this cipher, or any symmetric cipher, is that the two people wanting to exchange messages must meet beforehand to exchange a key; they would not want to increase the chances of someone else intercepting the key by exchanging it via telephone, letter, or e-mail. For two people on opposite sides of the world, or opposite sides of the United States, this is a great inconvenience.

2.1 Public Key Cryptography: RSA Encryption

The form of cryptography that will be discussed here and is of main interest, is public key cryptography. This is a way to exchange and share information publicly while keeping that information a secret from anyone other than the intended sources.

The theoretical basis for public key cryptography was laid by Whitfield Diffie, Martin Hellman, and Ralph Merkle in June 1976 at the National Computer Conference [15, p. 267]. Their system allowed for two people to “establish a secret via public discussion.” These men had introduced the idea of an asymmetric cipher, but could not find one that worked. Then in August 1977, three men from MIT, Ron Rivest, Adi Shamir, and Len Adelman, introduced an asymmetric cipher that worked [15, p. 274, 278]. Their cipher is known as RSA encryption, or the RSA algorithm for encryption. The important thing about using an asymmetric cipher is that two people do not have to meet beforehand to exchange keys. There exist two keys, a private key and a public key, each being very hard to obtain from the other. Anyone may have access to the public key, but the private key is kept secret. When one is used for encryption, the other is used for decryption. This, along with more information about the RSA algorithm, will be discussed later in more detail.

The ideas behind public key cryptography are certain techniques “based on numerical tricks used in a carefully crafted series of arithmetic computations” [16, p. 199]. Numerical values are chosen arbitrarily and are combined in such a way that is very difficult to reverse the process. Everyone has access to someone’s public key, but each individual has a unique private key corresponding to their public key. The initial values that are chosen make up the private key, while the combination of those same numbers becomes the public key.

Scenario 1 and 2, introduced in the beginning of this chapter, are just two uses for RSA encryption. The RSA algorithm produces public and private keys that are tied together by a series of mathematical computations, and either key can be used for encryption.

Using the public key for encryption is the form of RSA that we are most interested in. This is because encryption performed with the public key ensures secrecy of the message, since no one other than the person having the private key can decrypt it. That is not to say that it is not possible for someone to intercept the message, break the code, and decrypt it, but that would take an enormous amount of time [14, p.146]. Note that using the public key for encryption does not ensure authorship of the message since anyone with access to the public key can send it.

Encrypting a message with the private key ensures authorship of the message, but not secrecy. Secrecy is not ensured because any person with the corresponding public key could decrypt a message encrypted by the private key. Authorship is ensured because the private key provides a specific digital signature with the message which only occurs when it is encrypted using a specific private key. Thus each person has a specific digital signature tied to their private key. We will discuss more about digital signatures later. [16, p. 203–204].

RSA encryption is based on the idea that multiplying large primes together is relatively easy, but reversing the process to obtain the original numbers is close to impossible. First, two very large primes, each around 200 digits or more, are chosen. These two primes, p and q , are multiplied together to get the number n . The larger p and q are, the more difficult it will be for someone to try to factor n . The public key consists of n and an exponent e , which is relatively prime to $(p - 1)(q - 1)$. The private key consists of n and d , which is the inverse of $e \pmod{(p - 1)(q - 1)}$.

For the process of encryption, after establishing the keys, the intended message must be translated into numerical equivalents, referred to as m . Each letter has a corresponding number which can be found using the ASCII map. This map is a way the computer stores letters according to the ASCII character set. The message to be sent is encrypted using the formula $C = m^e \pmod{n}$. To decipher the sent message C , the formula $m = C^d \pmod{n}$ is used.

Since p and q are chosen to be so large, their product, n , could be 400 digits or more. So until someone finds a method that quickly factors n to give p and q , the RSA algorithm is a safe form of encryption. Considering no one has been able to find such a method for over 20 years, it seems that this algorithm will be the public key technique of choice for years to come.

2.2 The Mathematics Behind RSA

There are two main theorems that make up the encryption and decryption processes. They are the Chinese Remainder Theorem and Fermat's Little Theorem. The statements of each of these theorems, their proofs, and examples are below. Later the way these two theorems are incorporated into the RSA algorithm will be shown.

2.2.1 The Chinese Remainder Theorem

Before the statement of the Chinese Remainder Theorem is given, some definitions and notation must be given.

The symbol for divides is “ $|$ ”. For example, writing $a|b$ means that a divides b with a remainder of zero. When a does not divide b , we write $a \nmid b$.

Let \mathbb{Z}_m denote the m distinct equivalence classes of the relation congruence modulo m on \mathbb{Z} , ie, $a \equiv b \pmod{m}$ if and only if $m|(b - a)$ or if and only if $m|(a - b)$. Thus two integers lie in the same equivalence class if and only if they differ by an integral multiple of m . For an element $a \in \mathbb{Z}$, we will let \bar{a} denote the equivalence class with respect to congruence modulo m .

Theorem 2.1 (The Chinese Remainder Theorem). *Let $m_1, m_2, m_3, \dots, m_n$ be pairwise relatively prime positive integers. The system*

$$x \equiv a_1 \pmod{m_1}$$

$$x \equiv a_2 \pmod{m_2}$$

\vdots

$$x \equiv a_n \pmod{m_n}$$

has a unique solution modulo $m = m_1 m_2 m_3 \cdots m_n$. (That is, there is a solution x with $0 \leq x < m$, and all other solutions are congruent modulo m to this solution.)

Before the Chinese Remainder Theorem can be proved, some lemmas and theorems need to be introduced for the benefit of our understanding. First the Division Algorithm and the Euclidean Algorithm for finding the GCD will be stated, which are needed for Bezout’s Equation. The proofs of these two theorems will not be given since we are only interested in how they are used computationally. Proofs for these two theorems may be found in the following:

· The Division Algorithm can be found in *Numbers and Symmetry: An Introduction*

to Algebra by Bernard L. Johnston and Fred Richman, 1977 by CRC Press LLC.
· The Euclidean Algorithm can be found in *Abstract Algebra and Solution by Radicals* by John E. Maxfield and Margaret W. Maxfield, 1992 by Dover Publications, Inc.

Theorem 2.2 (Division Algorithm). *If $a \in \mathbb{Z}$ and $b \in \mathbb{P}$, then there exist unique integers q and r such that*

$$a = qb + r \text{ with } 0 \leq r < b. \text{ (Here } q \text{ is the "quotient" and } r \text{ is the "remainder".)}$$

Theorem 2.3 (Euclidean Algorithm for Finding GCD). *Suppose a and b are two integers with $0 < b \leq a$.*

Step 1: Apply the Division Algorithm to obtain q_1 and r_1 such that $a = q_1b + r_1$ with $0 \leq r_1 < b$.

If $r_1 = 0$, then b is the GCD of a and b .

Step 2: If $r_1 \neq 0$, then apply the Division Algorithm to obtain q_2 and r_2 such that $b = q_2r_1 + r_2$ with $0 \leq r_2 < r_1$.

If $r_2 = 0$, then the GCD of a and b is r_1 .

Step k : If $r_{k-1} \neq 0$, then divide r_{k-2} by r_{k-1} to obtain q_k and r_k such that: $r_{k-2} = q_k r_{k-1} + r_k$ with $0 \leq r_k < r_{k-1}$.

Since $r_1 > r_2 > \dots > r_k > 0$, we must eventually reach a remainder of 0 (in at most n steps). If r_n is the first zero remainder, then the GCD of a and b is r_{n-1} (ie., the last positive remainder) [10, p. 10].

Bezout's Equation will now be stated and a detailed example is given, which should suffice as a proof. The example will highlight how the Euclidean Algorithm is used in Bezout's Equation.

Theorem 2.4 (Bezout's Equation). *If a and b are positive integers, then there exist integers s and t such that $\gcd(a, b) = sa + tb$.*

As an example, two integers s and t will be found such that $\gcd(250, 198) = 250s + 198t$.

First, write the larger number, 250, as a linear combination of 198 and another number which must be found.

$$250 = 1 \cdot 198 + 52 \tag{2.1}$$

Continue this process by taking the larger of the two numbers used to write 250 as a linear combination and writing it as a linear combination of the smaller number and another number. In doing this, the $\gcd(250, 198)$ will be found. In the next step, 198 will be written as a linear combination of 52 and some other number. The process is completed when a final number can be written as a linear combination of another number and 0.

$$198 = 3 \cdot 52 + 42 \tag{2.2}$$

$$52 = 1 \cdot 42 + 10 \tag{2.3}$$

$$42 = 4 \cdot 10 + 2 \tag{2.4}$$

$$10 = 5 \cdot 2 + 0 \tag{2.5}$$

By the 4th division, write the $\gcd(250, 198) = 2$ as a linear combination of 42 and 10.

$$2 = 42 - 4 \cdot 10.$$

and by the 3rd division, write 10 as:

$$10 = 52 - 1 \cdot 42.$$

By substituting the expression for 10 into the previous equation, write 2 as a linear

combination of 52 and 42.

$$2 = 1 \cdot 42 - 4 \cdot 10 \tag{2.6}$$

$$= 1 \cdot 42 - 4(52 - 1 \cdot 42) \tag{2.7}$$

$$= 1 \cdot 42 - 4 \cdot 52 + 4 \cdot 42 \tag{2.8}$$

$$= 5 \cdot 42 - 4 \cdot 52 \tag{2.9}$$

Now use the 2nd division to write the $\gcd(250, 198) = 2$ as a linear combination of 198 and 52.

$$2 = 5 \cdot 42 - 4 \cdot 52 \tag{2.10}$$

$$= 5(1 \cdot 198 - 3 \cdot 52) - 4 \cdot 52 \tag{2.11}$$

$$= 5 \cdot 198 - 15 \cdot 52 - 4 \cdot 52 \tag{2.12}$$

$$= 5 \cdot 198 - 19 \cdot 52. \tag{2.13}$$

Finally, write 2 as a linear combination of 250 and 198, by the 1st division, which is the final step in finding s and t .

$$2 = 5 \cdot 198 - 19 \cdot 52 \tag{2.14}$$

$$= 5 \cdot 198 - 19(1 \cdot 250 - 1 \cdot 198) \tag{2.15}$$

$$= 5 \cdot 198 - 19 \cdot 250 + 19 \cdot 198 \tag{2.16}$$

$$= 24 \cdot 198 - 19 \cdot 250. \tag{2.17}$$

Hence, $s = -19$ and $t = 24$ and the $\gcd(250, 198) = 2 = -19 \cdot 250 + 24 \cdot 198$.

Despite the fact that there are more efficient methods of writing the $\gcd(a, b)$ as a linear combination of a and b , using the method in the above example will show that there is a unique factorization for any positive integer which can be factored into primes, where the primes are written in nondecreasing order [14, p. 138].

The next two lemmas concern divisibility. The second is a generalization of the first in the proof of the uniqueness of prime factorization.

Lemma 2.5. *If a , b , and c are positive integers such that $\gcd(a, b) = 1$ and $a|bc$, then $a|c$.*

Proof. Since $\gcd(a, b) = 1$, by Bezout's Equation there are integers s and t such that $sa + tb = 1$.

Multiplying both sides of this equation by c , we obtain $sac + tbc = c$.

Now using the theorem on the basic properties of the divisibility of integers, we can use the previous equation to show that $a|c$. The properties are:

- (1) If $a|b$ and $a|c$, then $a|(b + c)$;
- (2) If $a|b$, then $a|bc$ for all integers c ;
- (3) If $a|b$ and $b|c$, then $a|c$.

Thus by part (2), $a|tbc$. Since $a|sac$ and $a|tbc$, by part (1), we conclude that $a|(sac + tbc)$, and hence $a|c$, which is what we wanted to prove. \square

Lemma 2.6. *If p is a prime and $p|a_1a_2\dots a_n$ where each a_i is an integer, then $p|a_i$ for some i .*

Proof of uniqueness of the prime factorization of a positive integer. Suppose that the positive integer n can be written as the product of primes in two different ways, $n = p_1p_2\dots p_s$ and $n = q_1q_2\dots q_t$,

where p_i and q_j are primes such that $p_1 \leq p_2 \leq \dots \leq p_s$ and $q_1 \leq q_2 \leq \dots \leq q_t$.

Removing the common primes from the two factorizations gives:

$$p_{i_1}p_{i_2}\dots p_{i_u} = q_{j_1}q_{j_2}\dots q_{j_v},$$

where no primes occur on both sides of this equation and u and v are positive integers.

By Lemma 2.6 it follows that $p_{i_k} | q_{j_k}$ for some k . Since no prime divides another prime, this cannot happen. Thus, there can be at most one factorization of n into primes in nondecreasing order. \square

The significance of the following theorem is that it allows the division on both sides of a congruence by an integer that is relatively prime to the modulus.

Theorem 2.7. *Let m be a positive integer and let a , b , and c be integers. If $ac \equiv bc \pmod{m}$ and $\gcd(c, m) = 1$, then $a \equiv b \pmod{m}$.*

Proof. Since $ac \equiv bc \pmod{m}$, $m | (ac - bc) = c(a - b)$.

By Lemma 2.5, since $\gcd(c, m) = 1$, it follows that $m | (a - b)$. Thus we conclude that $a \equiv b \pmod{m}$. \square

The following theorem ensures us that an inverse of a positive integer $a \pmod{m}$ exists whenever a and m are relatively prime. The inverse is denoted by a^{-1} .

Theorem 2.8. *If a and m are relatively prime integers and $m > 1$, then an inverse of a modulo m exists. Furthermore, this inverse is unique modulo m . (That is, there is a unique positive integer a^{-1} less than m that is an inverse of a modulo m and every other inverse of a modulo m is congruent to a^{-1} modulo m .)*

Proof. By Bezout's Equation, since $\gcd(a, m) = 1$, there are unique integers s and t such that $sa + tm = 1$.

This implies that $sa + tm \equiv 1 \pmod{m}$.

Since $tm \equiv 0 \pmod{m}$, it follows that $sa \equiv 1 \pmod{m}$.

Consequently, s is an inverse of a modulo m . That this inverse is unique modulo m is shown by Theorem 2.7. Let a, b, c be integers and $c \neq 0$. Also, let m be a positive integer, such that $\gcd(c, m) = 1$. If $ac \equiv bc \pmod{m}$, then since c and m are relatively prime,

$$a \equiv b \pmod{m}.$$

Therefore the inverse is unique modulo m .

□

As an example of this theorem, the inverse of 4 modulo 9 will be found. The process in which this is solved is similar to the example done for Bezout's Equation, only here we will be working backwards.

Since $\gcd(4, 9) = 1$, Theorem 2.7 says that an inverse of 4 (mod 9) exists. Using the Euclidean Algorithm we see:

$$9 = 2 \cdot 4 + 1.$$

From this we can rewrite the equation to obtain:

$$-2 \cdot 4 + 1 \cdot 9 = 1.$$

Thus -2 is an inverse of 4 (mod 9).

Now we have sufficient information in order to prove the Chinese Remainder Theorem.

Proof of the Chinese Remainder Theorem. What we are going to do first is show that a solution exists by describing a way to construct the solution. Then we will show that the solution is unique modulo m .

In order to construct a simultaneous solution, let $M_k = \frac{m}{m_k}$, for $k = 1, 2, 3, \dots, n$. This means that M_k is the product of all the moduli except for m_k . Since m_i and m_k have no common factors greater than 1 when $i \neq k$, it follows that $\gcd(m_k, M_k) = 1$. Thus, by Theorem 2.8 we know that there exists an integer y_k , which is an inverse of $M_k \pmod{m_k}$, such that

$$M_k y_k \equiv 1 \pmod{m_k}.$$

To construct a simultaneous solution, form the sum

$$x = a_1M_1y_1 + a_2M_2y_2 + \cdots + a_nM_ny_n.$$

Now we will show that x is a simultaneous solution. First, note that since $M_j \equiv 0 \pmod{m_k}$ whenever $j \neq k$, all terms except the k th term in this sum are congruent to 0 $\pmod{m_k}$. Since $M_ky_k \equiv 1 \pmod{m_k}$, we see that $x \equiv a_kM_ky_k \equiv a_k \pmod{m_k}$, for $k = 1, 2, 3, \dots, n$.

Thus we have shown that x is a simultaneous solution to the n congruences.

Now we will show that x is unique modulo m .

If another integer x_1 satisfied the same system of congruences, then

$$x \equiv x_1 \pmod{m_1}$$

$$x \equiv x_1 \pmod{m_2}$$

\vdots

$$x \equiv x_1 \pmod{m_n}.$$

This can be rewritten as:

$$m_1|x - x_1$$

$$m_2|x - x_1$$

\vdots

$$m_n|x - x_1.$$

Hence, $m|x - x_1$, where $m = m_1m_2m_3 \cdots m_n$.

Therefore, $x \equiv x_1 \pmod{m}$,

and so x and x_1 are congruent modulo m_k , for $k = 1, 2, 3, \dots, n$. Thus we have shown that there exists one unique solution to the congruences. \square

The following is an example of the Chinese Remainder Theorem from the Chinese mathematician Sun-Tsu. He asked:

There are certain things whose number is unknown. When divided by 3, the remainder is 2, when divided by 5, the remainder is 3, when divided by 7, the remainder is 2. What will be the number of things?

This can be written out in the following way:

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 5 \pmod{7}.$$

To solve this system of congruences above, let $m = 3 \cdot 5 \cdot 7 = 105$, $m_1 = \frac{m}{3} = 35$, $m_2 = \frac{m}{5} = 21$, $m_3 = \frac{m}{7} = 15$. From this we see that 2 is an inverse of $m_1 = 35 \pmod{3}$, since $35 \equiv 2 \pmod{3}$; 1 is an inverse of $m_2 = 21 \pmod{5}$, since $21 \equiv 1 \pmod{5}$; and 1 is an inverse of $m_3 = 15 \pmod{7}$. Thus the solutions to this system are those x such that:

$$x = a_1 m_1 y_1 + a_2 m_2 y_2 + a_3 m_3 y_3 \tag{2.18}$$

$$= (2 \cdot 35 \cdot 2) + (3 \cdot 21 \cdot 1) + (2 \cdot 15 \cdot 1) \pmod{105} \tag{2.19}$$

$$= 233 \equiv 23 \pmod{105}. \tag{2.20}$$

It follows that 23 is the smallest positive integer that is a simultaneous solution. Thus we conclude that 23 is the smallest positive integer that leaves a remainder of 2 when divided by 3, a remainder of 3 when divided by 5, and a remainder of 2 when divided by 7 [14, p. 141-143].

2.2.2 Fermat's Little Theorem

Fermat's Little Theorem is the second of two theorems needed for the encryption and decryption processes to work. Before stating or proving the main theorem, some notations and definitions need to be given.

An element $\bar{a} \in \mathbb{Z}_m$ has a multiplicative inverse if and only if a and m are relatively prime (recall Theorem 2.8). Any element in \mathbb{Z}_m having a multiplicative inverse is called a *unit*.

For example, finding the multiplicative inverses of each nonzero element in \mathbb{Z}_7 , we have:

$$1^{-1} = 1 \quad \bar{2}^{-1} = \bar{4} \quad \bar{3}^{-1} = \bar{5}$$

$$\bar{4}^{-1} = \bar{2} \quad \bar{5}^{-1} = \bar{3} \quad \bar{6}^{-1} = \bar{6}$$

For a positive integer m , $\phi(m)$ is the number of positive integers less than or equal to m that are relatively prime to m . The function ϕ is called the *Euler phi function*. For example,

$$\phi(1) = 1 \quad \phi(2) = 1 \quad \phi(3) = 2$$

$$\phi(4) = 2 \quad \phi(5) = 3 \quad \phi(6) = 2$$

For a prime p , $\phi(p) = p - 1$.

Now Euler's Theorem can be stated. The proof will not be given since this theorem is so closely related to Fermat's Little Theorem, in which a proof is given below.

Theorem 2.9 (Euler's Theorem). *If a and m are relatively prime, then $a^{\phi(m)} \equiv 1 \pmod{m}$.*

Theorem 2.10 (Fermat's Little Theorem). *If p is a prime, then*

$$a^{p-1} \equiv 1 \pmod{p} \text{ if } p \nmid a, \text{ and}$$

$$a^p \equiv a \pmod{p} \text{ for every } a \in \mathbb{Z}.$$

Proof. Look at the product $1 \cdot 2 \cdot 3 \cdot \dots \cdot (p-1) \pmod{p}$. Now compare it with the product $(1a) \cdot (2a) \cdot (3a) \cdot \dots \cdot ((p-1)a)$. Both of these products have the same value, since all of the numbers in the first list appear in the second list, but in a different

order (i.e. (ka) is one of $1 \cdot 2 \cdot 3 \cdot \dots \cdot (p - 1)$, and (ka) is not (la) if $k \neq l$). Now divide both sides of the equation by $1 \cdot 2 \cdot 3 \cdot \dots \cdot (p - 1)$ to get $a^{p-1} \equiv 1 \pmod{p}$.

□

Here are some examples of Fermat's Little Theorem.

For $p = 5$, we see that $2^{5-1} = 2^4 = 16 \equiv 1 \pmod{5}$.

It is also true that $2^5 = 32 \equiv 2 \pmod{5}$.

Some composite integers have been found that satisfy $2^{p-1} \equiv 1 \pmod{p}$. Numbers such as these are called *pseudoprimes*. For example,

$p = 341 = 11 \cdot 13$, satisfies the equation, $2^{341-1} = 2^{340} \equiv 1 \pmod{341}$.

A pseudoprime p can exist when the following two conditions are met:

- 1) p is not a prime
- 2) for every $1 \leq a \leq p$, $a^{p-1} \equiv 1 \pmod{p}$.

To determine as nearly as possible whether or not a given number is prime, the number can be put through a series of tests on the computer called *probabilistic primality tests*. "The probability that an integer that passes a series of tests is prime is close to 1" [14, p. 146]. These tests are extremely helpful for RSA encryption because they enable one to quickly find the 2 large primes, p and q , needed to produce a usable key. Despite the small amount of time needed to find p and q , the time it takes to factor n , their product, is a tremendously unreasonable amount of time.

2.2.3 The Encryption and Decryption Processes

Now it will be shown how Fermat's Little Theorem and the Chinese Remainder Theorem are used in the processes of RSA encryption and decryption.

"If $de \equiv 1 \pmod{(p-1)(q-1)}$, then there exists an integer k such that $de = 1 + k(p-1)(q-1)$.

Then $C^d = ((m^e)^d) = m^{ed} = m^{1+k(p-1)(q-1)}$.

By Fermat's Little Theorem (assuming $\gcd(m, p) = \gcd(m, q) = 1$), it follows that $m^{p-1} \equiv 1 \pmod{p}$ and $m^{q-1} \equiv 1 \pmod{q}$. Therefore

$C^d \equiv m \cdot (m^{p-1})^{k(q-1)} \equiv m \cdot 1 \equiv m \pmod{p}$ and $C^d \equiv m \cdot (m^{q-1})^{k(p-1)} \equiv m \cdot 1 \equiv m \pmod{q}$.

Since $\gcd(p, q) = 1$, it follows from the Chinese Remainder Theorem that

$C^d \equiv m \pmod{pq}$ " [14, p.147].

The following is an example of the RSA encryption and decryption processes [15, p. 379-381]. The calculations for this example were done using Mathematica 4.0. For very small numbers, a scientific or graphing calculator would suffice.

Susan picks two large prime numbers, p and q , which she will keep secret. Normally each number would be around 200 digits to ensure safety, but for the sake of our example we will choose smaller numbers. The numbers are $p = 2129$ and $q = 3041$. Multiplying p and q together, Susan gets another number, n , which is 6474289. Now she picks a number e which must be relatively prime to $(p-1)(q-1)$; here it is chosen to be 213. Later we will see the importance of its being relatively prime.

Now Susan can publish her public key, which consists of e and n , in something like a telephone directory. Since these two numbers are needed for encryption, they must be available to anyone who may want to encrypt a message and send it to her.

To encrypt a message, the message must first be converted into a number, m . Normally the letters of the words are changed into numbers using the ASCII map, but for our example we will simply send her a numerical message. After obtaining m , it is encrypted to give the ciphertext, C , using the formula

$$C = m^e \pmod{n}.$$

Now imagine that David wants to send Susan the message 2577, which is their special code for meeting at a specific time. Thus, $m = 2577$. To encrypt this message, David begins by looking up Susan's public key, and sees that $n = 6474289$ and $e = 213$. Using the encryption formula, he finds C to be:

$$C = 2577^{213} \pmod{6474289} = 4737942.$$

Now David sends the ciphertext, $C = 4737942$, to Susan. We know that exponentials in modular arithmetic are one-way functions, hence the reason for the algorithm's asymmetry. Thus, it is very difficult to work backwards from C and recover the original message, m . A third person could not easily decipher the message.

Once Susan receives the message, she can decipher it using the decryption key. She calculates d , part of her private key, in the following way:

$$ed = 1 \pmod{(p-1)(q-1)} \tag{2.21}$$

$$213d = 1 \pmod{(2128)(3040)} \tag{2.22}$$

$$d = 3067517. \tag{2.23}$$

Now it is clear why e is picked to be relatively prime to $(p-1)(q-1)$.

In order to decrypt the message, Susan uses the following formula:

$$m = C^d \pmod{n} \tag{2.24}$$

$$= 4737942^{3067517} \pmod{6474289} \tag{2.25}$$

$$= 2577, \tag{2.26}$$

which was the original message sent.

2.3 Digital Signatures

Scenario:

Suppose a principal sends a message to teachers and students saying school is canceled. The teachers, wanting to make sure that the message is not a prank, need a way of verifying its authenticity. If the principal did indeed send the message, it will contain his specific digital signature so that anyone with his public key can decrypt the message and see whether or not it was sent by the principal. If it was not sent by him, then after decryption, the message would become mixed-up nonsense.

Digital signatures are tied to one specific private key, so only the person with the private key can produce the corresponding digital signature. If a person is able to decrypt a message with a public key, then only the corresponding private key can be used to encrypt it. For example, suppose someone wants to forge a message. If the sender uses a person's public key to encrypt the message, then when the recipient tries to decrypt it using the public key, it will make no sense. This is because the message was not encrypted using the private key. Digital signatures can be used when transferring funds or depositing money into the bank over the computer. Instead of signing a deposit slip, the digital signature acts as your written signature.

An example of this is the following:

Suppose the same message as before, 2577, was to be sent, but this time it is encrypted using the private key, $n = 6474289$ and $d = 3067517$, to ensure authorship. After receiving the encrypted message, Susan would decrypt it using the corresponding public key, consisting of $e = 213$ and n . The only drawback from this process is that it does not provide security since anyone with access to the corresponding public key can decrypt the message.

Here the process for using digital signatures is shown:

The message being sent is $k = 2577$. The following formula is used to obtain the encrypted message, l :

$$l = k^d \pmod{n} = 2577^{3067517} \pmod{6474289} = 1003691.$$

Upon receiving the message l , the recipient would then decrypt the message in the following way:

$$k = l^e \pmod{n} = 1003691^{213} \pmod{6474289} = 2577,$$

which is the correct message.

Conclusion

In this thesis we have discussed the benefits of including discrete mathematics in the secondary-school curriculum, and have focused on two well-known topics of discrete mathematics. These topics are only two of the many that discrete mathematics has to offer both students and teachers. One of the great qualities about these topics is the variability of their degree of difficulty. This is helpful for two reasons. First, it makes these topics available to all skill levels. Second, the problems are able to hold the attention of students by becoming more difficult over time. Therefore, these topics have the potential to interest every student. Since the problems in discrete mathematics relate well to the real world, students are able to understand and relate easily to the problems.

Teaching discrete mathematics alongside the traditional curriculum provides students with different methods for learning the material and can help teachers reach out to every student. Discrete mathematics also gives students a better overall view of what mathematics has to offer and can introduce students to all the different forms of mathematics that exist. Mathematics does not only consist of concepts proved long ago by Euler and Fermat; it is forever expanding, with new concepts being discovered and proved all the time.

The Traveling Salesman Problem is an excellent topic for secondary students because of all the various forms of it that exist. The TSP can be adapted to take on whatever form or shape one desires, according to the degree of difficulty appropriate for one's students.

RSA encryption is an interesting topic for students. It enables them to create keys and send messages to one another. While this topic could be simple and involve

only calculators, it can be made more difficult to involve the use of a computer.
Either way, this and the TSP are topics for all skill levels.

References

- [1] Bogart, K.P., The Roles of Finite and Discrete Mathematics in College and High School Mathematics, in *Discrete Mathematics Across the Curriculum, K-12* (eds. M.J. Kenney, and C.R. Hirsch), NCTM, Inc, Reston, Virginia; 1991, p. 83.
- [2] Buydos, J.F. with Khan, D., *Cryptology*, American Cryptogram Association, Vernon Hills, Illinois; January 1997.
- [3] Dossey, J.A., Discrete Mathematics: the Math for Our Time, in *Discrete Mathematics Across the Curriculum, K-12* (eds. M.J. Kenney, and C.R. Hirsch), NCTM, Inc, Reston, Virginia; 1991, pp. 1-5.
- [4] Hart, E.W., Discrete Mathematics: An Exciting and Necessary Addition to the Secondary School Curriculum, in *Discrete Mathematics Across the Curriculum, K-12* (eds. M.J. Kenney, and C.R. Hirsch), NCTM, Inc, Reston, Virginia; 1991, pp. 67-76.
- [5] Hart, E., Maltas, J., and Rich, B., Teaching Discrete Mathematics in Grades 7-12, *Mathematics Teacher* **83** (ed. H.L. Schoen), May 1990, pp. 362, 364.
- [6] Herstein, I.N., *Abstract Algebra*, Macmillan Publishing Company, New York; 1986, p. 72.
- [7] Holliday, R.L., Graph Theory in the High School Curriculum, in *Discrete Mathematics Across the Curriculum, K-12* (eds. M.J. Kenney, and C.R. Hirsch), NCTM, Inc, Reston, Virginia; 1991, p. 94.
- [8] Ireland, K. and Rosen, M., *A Classical Introduction to Modern Number Theory*, Springer-Verlag New York, Inc, 1990, p. 33.
- [9] Johnson, D.S. and Papadimitriou, C.H., Computational Complexities, in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* (eds. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Khan, and D.B. Shmoys); John Wiley & Sons Ltd, 1985, pp. 43-44.
- [10] Lax, R.F., *Modern Algebra and Discrete Structures*, HarperCollins Publishers, New York, New York; 1991, pp. 10, 146-148.
- [11] Miller, D.W., Discovering and Applying Euler's Formula, in *Discrete Mathematics Across the Curriculum, K-12* (eds. M.J. Kenney, and C.R. Hirsch), NCTM, Inc, Reston, Virginia; 1991, pp.96-103.

- [12] Picker, S., Using Discrete Mathematics to Give Remedial Students a Second Chance, in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **36** (eds. J.G. Rosenstein, D.S. Franzblau, and F.S. Roberts), American Mathematical Society; 1997, pp. 35–40.
- [13] Roberts, F.S., The Role of Applications in Teaching Discrete Mathematics, in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **36** (eds. J.G. Rosenstein, D.S. Franzblau, and F.S. Roberts), American Mathematical Society; 1997, pp. 106–107.
- [14] Rosen, K.H., *Discrete Mathematics and Its Applications*, AT & T, 1999.
- [15] Singh, S., *The Code Book: The Evolution of Secrecy from Mary Queen of Scots to Quantum Cryptography*, Doubleday, New York, 1999.
- [16] Smith, R.E., *Internet Cryptography*, Addison Wesley Longman, Inc, Reading, Massachusetts; 1997.
- [17] West, D.B., *Introduction to Graph Theory*, Prentice-Hall, Inc, Upper Saddle River, New Jersey; 1996.
- [18] www.math.princeton.edu/tsp

Vita

Aimee Beth Boyd was born on July 25, 1977, in Corpus Christi, Texas. She finished her undergraduate studies at Auburn University in June 1999. In August 1999 she came to Louisiana State University to pursue graduate studies in mathematics. She is currently a candidate for the degree of Master of Science in mathematics, which will be awarded in May 2002.