Mathematics of Information Processing and the Internet: Essential Mathematics in a 21st Century High School Curriculum

Eric W. Hart

A shorter version of this paper is published in Mathematics Teacher (in press), Reston, VA: National Council of Teachers of Mathematics.

Abstract

The mathematics of information processing and the Internet can be organized around four fundamental themes – access, security, accuracy, and efficiency. This mathematics is vital in the modern, technology-rich, information-dense world in which students live. By including appropriate topics in the high school curriculum, students will acquire the quantitative literacy they need and they will keep open doors of opportunity for college and the world of work.

Introduction

The Internet is everywhere in contemporary society. It is emblematic of the information age in which we live. We are inundated with information that we must effectively process so that it is manageable and useful.

Think about buying a song online. First you must find the song. Suppose you know that the style of music is electronica and that the artist’s name contains the word “Mouse,” but you can’t remember any more information about the song. You might google “mouse” to look for the song, but you will get too many results that don’t have anything to do with music. It would be more efficient to search for “electronica AND mouse.” For many search engines, AND is the default setting, so you can just search for “electronica mouse.” When you look at the list of results you see the name of the artist you like – Mouse on Mars. Then you can continue to find the particular song you want. What does this have to do with mathematics? Internet searches use set theory and logic. For example, the search for “electronica mouse” uses the set operation of intersection and the Boolean (or logical) operator AND.

Once you find the song, you can buy it and download it. When you buy the song using a credit card, you want to be sure that the credit card number is kept secure and is not stolen by some Internet prowlers. Ensuring credit card security requires the mathematics of cryptography. After purchasing the song, you might download it rather than have it mailed. You would like the download to be fast and accurate. Also, it would be nice if the music file is not so large that it takes up too much space yet large enough so that the musical qualities are preserved. Ensuring an
accurate download involves the mathematics of error-detecting and correcting codes. To make the download fast and the file size compact requires the mathematics of data compression.

This example illustrates four key themes of information processing, particularly as related to the Internet – access (finding information easily), security (keeping information confidential), accuracy (ensuring accurate information), and efficiency (data compression). In this article, each theme will be briefly discussed with reference to high school mathematics.

**Access**

To be useful, information must be accessible. There are several ways mathematics is applied to make information more accessible. Consider databases. Relational databases store information in data tables. Each row of a table can be thought of as an \( n \)-tuple, and thus the data tables can be thought of as sets of \( n \)-tuples, that is, relations. Information is retrieved and reorganized using relational algebra, whereby queries, built from relational operators such as select, project, and join, are used to form new tables (relations) from old. Thus, relational databases use mathematics to help provide access to information. Another way to make information stored on computers accessible is through searching and sorting algorithms, such as those studied in computer science.

An aspect of the issue of access in information processing that is more directly related to high school mathematics is searching using set theory and Boolean (or logical) operators. For example, a Quick Tip of the Week from Apple Hot News on October 21, 2008, stated that, “In addition to searching any Mac on your network, Spotlight also lets you take advantage of Boolean search operators—AND, OR, NOT—and other rich search vocabulary” (http://www.apple.com/business/theater/#tutorial=booleansearches).

Similarly, the Library of Congress website provides a help page devoted to Boolean searches to help online visitors find information in the library more easily, as shown in Figure 1.
**Figure 1: Boolean Searching online at the Library of Congress**
(Retrieved from catalog.loc.gov/help/Boolean.htm on 12/14/08)

This figure could be used as a starting point and real-world application for a lesson on elementary set theory, Venn diagrams, and basic logic, in which students learn about the set operations intersection, union, and set difference, and how these correspond to the logical operators AND, OR, and NOT, respectively.

Most Internet search engines use these operators, although the notation varies. On Google, you could search for information about Lennon and McCartney by entering “Lennon McCartney,” since Google inserts AND between search words by default. To search for information about Lennon or McCartney you would enter “Lennon OR McCartney.” To find information about Lennon but not McCartney you would use the search phrase “Lennon – McCartney,” since the symbol ‘–‘ is used for NOT on Google.

A more advanced example of how mathematics is used to make information accessible is Google’s proprietary method for ranking web pages. This method uses a matrix in which the $i-j$ entry is the reciprocal of the number of links on page $j$ if page $j$ links to page $i$, and 0 otherwise. Then methods of linear algebra are applied to produce the page ranking for a particular search. (See Austin 2006.)

Mathematics is vital to make information accessible. Much of the mathematics discussed here, while interesting and appropriate for some students, is not core mathematics for all students. However, elementary set theory and basic logic are important topics that all students should study. These topics are not only central to information processing and the Internet, they

<table>
<thead>
<tr>
<th>Concept</th>
<th>Search Examples</th>
<th>Retrieval Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>rogers AND hammerstein children AND poverty civil war AND virginia</td>
<td>Retrieves only records containing both terms.</td>
</tr>
<tr>
<td>OR</td>
<td>sixties OR 50s OR 1950s labor OR labour email OR e-mail OR “electronic mail”</td>
<td>Retrieves records containing either one or more terms.</td>
</tr>
<tr>
<td>NOT</td>
<td>caribbean NOT cuba jockey NOT disc “civil war” NOT american</td>
<td>Excludes records containing the second term.</td>
</tr>
<tr>
<td>NESTING</td>
<td>fruit AND (banana OR apple) (women OR woman) AND basketball (color OR colour) AND (decorate OR decoration) NOT (art OR architecture)</td>
<td>Use parentheses ( ) to group portions of boolean queries for more complex searches.</td>
</tr>
</tbody>
</table>
also have applications in many other areas of life and mathematics, and these topics should be part of the high school mathematics curriculum for all students.

**Security**

For information to be useful, it must be secure. Governments, companies, and individuals want sensitive information to be secure and private. You don’t want your credit card number stolen when you send it through a website to an online vendor. Embassies abroad need to securely send information back to their home governments. An email message sometimes must be sent securely so that the sender and receiver know that it is private and authentic. All of this is achieved through *cryptography*, the study of mathematical concepts and methods for making information secure.

Cryptography is used to design *cryptosystems* – systems for encrypting and decrypting information using keys. A cryptosystem works according to diagram in Figure 2.

![Diagram of a cryptosystem](image)

**Figure 2**: A cryptosystem uses keys to convert plaintext to ciphertext and then back to the original plaintext

There are two basic types of cryptosystems – symmetric-key and public-key. In a *symmetric-key cryptosystem*, the same key is used to encrypt and decrypt. Thus, the security of the system depends on the secrecy of the key. In a *public-key cryptosystem*, different keys are used for encryption and decryption. One key is made public, and the other is kept secret. Since symmetric-key systems are faster, but public-key systems are more secure, hybrid cryptosystems are often used, in which the same key is used to encrypt and decrypt but the key is transmitted from sender to receiver using a public-key system.

You often see cryptosystems in action when you use the Internet. For example, you might see a warning message such as on the left in Figure 3 when you are entering personal information on a Web page. When shopping online, you know that cryptography is being used to keep your transaction secure when you see that the website address begins with “https” instead of “http,” as on the right in Figure 3. This indicates that the Secure Sockets Layer (SSL) protocol is being used to securely transfer information. According to the Apple OS X Help guide, “Web browsers and many websites use the SSL protocol to transfer confidential user information, such
as credit card numbers. SSL uses a public and private key encryption system” (OS X 10.5.6 Help, “About Secure Sockets Layer”).

![Security Warning](https://www.amazon.com/gp/cart/)

![https://www.amazon.com](https://www.amazon.com/gp/cart/)

![https://www.paypal.com](https://www.paypal.com/)

Figure 3: Examples of cryptography in action on the Internet

A common example of a symmetric-key cryptosystem is a *substitution cipher*. In a substitution cipher, characters of the plaintext message are replaced by other characters to create the ciphertext. The most simple substitution ciphers are so-called *Caesar ciphers*, named in honor of the method Julius Caesar (100–44 B.C.) used to send confidential messages. Using this method, you simply replace each letter in a plaintext message with a letter that is a fixed number of places further down the alphabet. For lessons and an applet related Caesar ciphers, see the “Code Crackers” lesson and the “Codes” applet on the NCTM Illuminations website (http://illuminations.nctm.org). These ciphers are easy to break simply by using knowledge about the relative frequencies of letters in the English language.

A more secure substitution cipher is a *Hill cipher*. This type of cipher uses matrix multiplication to scramble the plaintext message, replacing the same letter with different letters at different places in the message. Decrypting then involves multiplying by the inverse matrix. (Note that this is still a symmetric-key system, with the “same” key used to encrypt and decrypt since by “same” we mean either exactly the same or that the decryption key is easy to directly calculate from the encryption key.) [St. John 1998] provides a thorough explanation of Hill ciphers that is suitable for high school students.

The biggest drawback to symmetric-key cryptosystems is that the common key must be transmitted and kept secure. Before I can send you a secret message, I need to send you the key we will use. But to send you the key securely, I’ll need to encrypt it with another key, which I’ll need to send to you, but then we’ll need a key for that key, and so on.

An elegant solution to this quandary was discovered in the mid-1970s with the development of public-key cryptography by Whitfield Diffie and Martin Hellman at Stanford...
University and Ronald L. Rivest, Adi Shamir, and Leonard Adleman at the Massachusetts Institute of Technology. (It appears that James Ellis, Clifford Cocks, and Malcolm Williamson developed the idea earlier as part of secret work for a British intelligence service, but this was not made public until the 1990s.) Public-key cryptography is one of the most significant developments in the history of cryptography.

Consider the RSA public-key cryptosystem (with initials in honor of the MIT developers). This system is based on the fact that it is relatively easy to multiply two large numbers, but difficult to factor a large composite number. Messages are first converted to numbers (e.g., A becomes 1, B becomes 2, etc.) and then the numbers are transformed using certain computations. The computations are done using modular arithmetic. Here’s how it works.

To begin, multiply two very large prime numbers, \( p \) and \( q \), to generate a large composite number, \( n \). Next, compute \( r = (p - 1)(q - 1) \). Then choose a number \( e \) (for encrypt) that has a multiplicative inverse, \( d \) (for decrypt), under multiplication mod \( r \). Encryption involves raising to the power \( e \) and reducing mod \( n \), while decryption involves raising to the power \( d \) and reducing mod \( n \). The numbers \( n \) and \( e \) provide the public encryption key available to anyone, and \( d \) is the private key used for decryption. This procedure works, that is, the decryption undoes the encryption, because of a special case of Euler’s Theorem: \((M^e)^d = M \mod n\), where \( p \) and \( q \) are prime numbers, \( n = pq \), \( r = (p - 1)(q - 1) \), and \( e \) and \( d \) are multiplicative inverses mod \( r \).

Suppose Alice wants to send a message to Bob using the RSA cryptosystem. (Alice and Bob have become the traditional communicators in cryptography. Usually the communicators are computers that have been programmed to implement a cryptosystem.) Bob, the receiver, has created public and private keys using the method above. He publishes the public key so that it is available to anyone who wants to send him a message. He keeps the private key secret, known only to himself. Alice uses Bob’s public key to encrypt her message and sends it to Bob. Bob uses his private key to decrypt the message.

Think about the security of this message transmission. Two very large prime numbers are multiplied to create the public encryption key, which is comprised of the numbers \( n \) and \( e \) from above. Suppose an adversary, Carol, wants to steal the message. Carol already knows \( n \) and \( e \), since these are publicly known as part of the public key. To steal the message, that is, decrypt it, Carol needs the private key, which is the number \( d \). To find \( d \), Carol needs to know \( r \), since \( d \) is the multiplicative inverse of \( e \mod r \). To find \( r \), Carol needs to know \( p \) and \( q \), since
\[ r = (p - 1)(q - 1). \] But \( p \) and \( q \) are the prime factors of \( n \), which Carol already knows. Thus, to break the code an adversary would need to factor the large composite number \( n = pq \). Since factoring large numbers is presumed to be a hard problem in mathematics, transmitting information using RSA public-key cryptography is presumed to be secure.

To say that factoring large numbers is presumed to be hard refers to problem complexity and how long it might take a computer to solve the problem. This directly relates to a famous unsolved problem in complexity theory about whether or not \( P = NP \), where \( P \) and \( NP \) are classes of problems at different complexity levels. (For further information about this see the Clay Millennium Problems website at \( \text{http://www.claymath.org/millennium/} \).)

How much of this mathematics is core content for all high school students? We want students to be literate consumers in an Internet-based information age; and we want to keep doors of opportunity open for them as we prepare them for college and the world of work. According to these criteria, it is reasonable to expect all students to learn the basics of modular arithmetic and, in general, how it is applied to provide information security. In addition to the fundamental application to information processing, this is rich mathematics that will help students deepen their understanding of algebraic structures and properties, and strengthen their reasoning and problem solving skills.

**Accuracy**

In addition to being accessible and secure, information must also be accurate to be useful. Error-detecting and -correcting codes help make information accurate. Generally, error-detecting codes are used to help ensure accuracy of identification numbers, like ZIP codes for mail delivery, UPC codes for consumer products, and ISBN numbers for books; while error-correcting codes are used to ensure accuracy when transmitting information, such as data from a deep-space probe, cell-phone conversations, and price information from a grocery store checkout scanner.

Error-detecting codes for identification (ID) numbers typically work by appending a check digit to the ID number. The check digit is chosen so that some modular arithmetic property is satisfied. For ZIP codes, the check digit is the 10th digit appended to a nine-digit ZIP code so that the sum of all ten digits is equivalent to 0 mod 10. For UPC codes, like 7-4236521685-5 for
a carton of heavy whipping cream (see Figure 4), the final digit is a check digit chosen so that when you add all the digits in odd positions, triple that sum, and then add the result to the sum of all the digits in the even positions, you get 0 mod 10. Different algorithms and modular systems are used for other ID numbers, such as mod 11 for ten-digit ISBN numbers, mod 9 for some traveler’s checks, mod 7 for many airline and rental car ID numbers, and mod 43 for some health industry codes.

An error in an ID number is detected if the designated algorithm does not meet the specified criterion. When that happens, the checker at the grocery store, for example, must enter the ID number by hand instead of simply scanning the product. Different codes have different error-detecting and -correcting capabilities, and no code detects and corrects all possible errors.

Codes used for ID numbers typically focus on detecting errors and are not particularly strong in correcting errors. On the other hand, codes used for transmitting information are designed to both detect and correct errors. A common type of error-correcting code used in digital data transmission is a linear code (also called a group code), which uses the ideas of Hamming distance and maximum-likelihood decoding.

In these codes, data is represented with binary code words, that is, strings of 0s and 1s. Each 0 or 1 is called a bit (binary digit). A set of binary strings is a linear code if it is closed under addition. That is, when you add two binary strings in the set bit by bit using mod 2 arithmetic, you obtain another binary string in the set. Figure 5 shows an example of a linear code with four code words.

```
Set of binary strings: {000, 011, 101, 110}
The sum of any two strings (bit by bit in mod 2) is another string in the set.
011 + 101 = 110
011 + 110 = 101
101 + 110 = 011
000 + any other string = that string
Any string + itself = 000
```

Figure 5: A linear code with four code words
The Hamming distance between two code words is the number of bits in which the two words differ. For example, the Hamming distance between 011 and 110 is two, since they differ in two bits—the first and third bits are different.

A message is encoded using the code words in the linear code. The four code words in the linear code in Figure 5 might represent four shades of gray in an image. When the image (message) is sent, some of the bits could get jumbled, that is, errors could occur. The received message is decoded using maximum-likelihood decoding, whereby each received string is decoded to a code word to which it is closest.

Suppose 010 is received. This is detected as an error, since 010 is not one of the code words in the set. The computer receiving the message will attempt to correct the error by decoding the received string to a code word that differs from it by the least number of bits. Thus, 010 could be decoded to 110, since this is a code word that differs from the incorrect string by one bit. However, the code word 011 also differs from the incorrect string by one bit. Therefore, this error cannot be corrected since it is not possible to specify a unique decoding.

To ensure a unique decoding, one must consider the minimum Hamming distance between all the code words. In the code in Figure 5, the minimum distance between any two code words is two. Figure 6 shows a linear code with minimum Hamming distance 3.

A key result is that if the minimum Hamming distance for a code is greater than or equal to 3, then the maximum likelihood decoding scheme corrects all single errors. Suppose the linear code in Figure 6 is used to send a message and the string 00111 is received. This is detected as an error since it is not the same as any of the four code words. Using maximum-likelihood decoding, the string decodes to the closest code word, 01111, which differs in one bit. No other code word differs in one bit, so this is a unique decoding and the error is thus corrected. Because
the minimum Hamming distance between any two code words is 3, any single-bit error will be corrected.

There is an effective procedure for generating single-error correcting linear codes (i.e., linear codes with minimum Hamming distance 3):

- Construct a matrix $H$, with entries that are 1s and 0s such that no column is all 0s and no two columns are equal.
- Find all solutions to the matrix equation $Hx = 0$, where $x$ is a one-column matrix.
- This set of solutions (which is technically the null space of the linear transformation represented by the matrix $H$) is a linear code with minimum Hamming distance 3.

The name linear code derives from the use of linear algebra in this process – linear transformations, null space, and solving a system of linear equations. (For more on this method and why it works, see Hirsch et al. 2009 and Malkevitch et al. 1991.)

While some of the mathematics of error-detecting and -correcting codes is beyond the scope of high school mathematics, much of it is accessible and appropriate for some high school students. The section at the end of this article includes some recommendations for important topics to consider in a high school mathematics curriculum.

**Efficiency**

A final issue to consider in information processing is efficiency. We will consider this in the context of data compression. Think about documents, software, photos, music, and video that you store on your computer or email to a friend or download from the Internet. You would like these digital data files to transmit efficiently, download quickly, and not take up too much storage space on your iPod or hard drive. This is achieved through data compression techniques. You know that data compression has occurred when you see files that end in .jpg (photos), .mp3 (music), .mpg (video), or .zip (general file compression).

The basic strategy of data compression is to use a variable-length code to encode more-frequently occurring data with shorter code words. Thus, $E$ is encoded with a shorter code word (binary string) than $Q$. In fixed-length codes, called block codes, such as the well-known ASCII code used to convert characters to 1s and 0s for computer use, each code word has the same number of bits. Using a block code, $E$ and $Q$ are each encoded with the same number of bits. By
using a variable-length code where E is represented with fewer bits than Q, messages will be compressed.

A variable-length code used to compress data needs to have certain properties. It must be uniquely decodable and not require any false starts or backtracking in the decoding process. The variable-length code in Figure 7a is not uniquely decodable – the string 0110 could be decoded as AEEA, ACA, or AEB. The code in Figure 7b is uniquely decodable, but could involve backtracking and false starts – the string 00111101111 decodes uniquely as ECBAB, but when you try to do the decoding you will likely have some false starts and need to backtrack a few times.

A Variable-Length Code

<table>
<thead>
<tr>
<th>character</th>
<th>code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>11</td>
</tr>
</tbody>
</table>

A Variable-Length Code that is Uniquely Decodable

<table>
<thead>
<tr>
<th>character</th>
<th>code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>01</td>
</tr>
<tr>
<td>C</td>
<td>011</td>
</tr>
<tr>
<td>B</td>
<td>111</td>
</tr>
</tbody>
</table>

A Variable-Length Code that is Uniquely Decodable and Requires No False Starts or Backtracking (Prefix-Free)

<table>
<thead>
<tr>
<th>character</th>
<th>code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>101</td>
</tr>
<tr>
<td>D</td>
<td>111</td>
</tr>
<tr>
<td>L</td>
<td>100</td>
</tr>
<tr>
<td>V</td>
<td>1101</td>
</tr>
<tr>
<td>K</td>
<td>1100</td>
</tr>
</tbody>
</table>

Figure 7a

Figure 7b

Figure 7c

The code in Figure 7c is uniquely decodable and requires no false starts or backtracking. Note that it has the property that no code word is the prefix of any other code word, in contrast to the codes in Figures 7a and 7b. This is what prevents the need for backtracking or false starts. Such codes are called prefix-free codes (or simply prefix codes).

One of the earliest data compression methods, which is still used as part of most modern proprietary compression algorithms, is a Huffman code. A Huffman code is a variable-length prefix-free code with the particularly nice property that it has the minimum average code word length among all prefix-free codes.
A Huffman code is constructed by first determining the relative frequency of all the characters in the designated message. Figure 8 shows the relative frequencies of six letters in a particular message to be sent.

The relative frequencies for each character become the bottom vertices of a tree (i.e., a vertex-edge graph that is connected and has no cycles). The tree, called a Huffman tree, is built up from these vertices by combining pairs of vertices with small relative frequencies.

You start with the pair having smallest frequencies and draw edges up from each to form a new vertex that is labeled with the sum of the frequencies. Continue this process until you have a single vertex at the top of the tree. The two vertices used at each step to create the new sum vertex are not used again in the process. See Figure 9 for a Huffman tree associated with the characters and relative frequencies shown in Figure 8. Note that a Huffman tree may not be unique, but all will yield a code the same average code word length. (For more about Huffman codes see Malkevitch and Froelich 1993.)

<table>
<thead>
<tr>
<th>character</th>
<th>relative frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0.1</td>
</tr>
<tr>
<td>C</td>
<td>0.1</td>
</tr>
<tr>
<td>D</td>
<td>0.15</td>
</tr>
<tr>
<td>F</td>
<td>0.2</td>
</tr>
<tr>
<td>G</td>
<td>0.2</td>
</tr>
<tr>
<td>H</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Figure 8: Relative frequencies

Figure 9: A Huffman tree for the relative frequencies in Figure 8

Once the Huffman tree is constructed, you find code words for each character represented by the vertices at the bottom of the tree as follows. Start at the top vertex and move down, vertex
by vertex, to a bottom vertex. Encode each step of this zig-zag, left-right path with a 0 for each left and a 1 for each right. This gives the code word for the character represented by the bottom vertex. The code words corresponding to the data and tree in Figures 8 and 9 are shown in Figure 10.

<table>
<thead>
<tr>
<th>Code</th>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1 (B)</td>
<td>000</td>
</tr>
<tr>
<td>0.1 (C)</td>
<td>001</td>
</tr>
<tr>
<td>0.15 (D)</td>
<td>100</td>
</tr>
<tr>
<td>0.2 (F)</td>
<td>101</td>
</tr>
<tr>
<td>0.2 (G)</td>
<td>01</td>
</tr>
<tr>
<td>0.25 (H)</td>
<td>11</td>
</tr>
</tbody>
</table>

**Figure 10: Code words for data and tree in Figures 8 and 9**

The final section of this article provides recommendations for which topics associated with the issue of efficiency (data compression) could be included in high school mathematics.

**Topics for a 21st Century High School Mathematics Curriculum**

The mathematics of information processing and the Internet can be organized around four fundamental themes – access, security, accuracy, and efficiency. This includes many interesting and important topics. Which topics should high school students study? Three filters can be used to help answer this question: (1) Which topics contribute to the quantitative literacy that all students need in the modern technology-rich, information-dense, “flat world” (cf. Friedman 2008) in which they live; (2) Which topics keep all doors of opportunity open for students as they move into college and the world of work; and (3) Which topics are not only fundamental to the mathematics of information processing, especially as related to the Internet, but also are valuable mathematical topics in their own right and have other important applications. Applying these three filters yields the following recommendations.
Essential topics for all students:

- Basic set theory, including the operations of union, intersection, set difference, and set complement, and the subset relation (related to the issue of access in information processing);
- Basic Boolean logic, including AND, OR, and NOT (related to the issue of access in information processing);
- Basic modular arithmetic, including congruence mod n, arithmetic mod n, and multiplicative inverses mod n (related to the issues of security and accuracy in information processing).

Valuable topics for many students:

- Use of inverse matrices in cryptography, as in Hill ciphers;
- Hamming distance, as an example of a metric in a setting other than traditional geometry;
- Basic ideas of data compression, including variable-length codes versus block codes and the fundamental strategy of encoding more-frequently occurring data with shorter code words.

Interesting topics for some students:

- Ideas of linear algebra applied to page rankings in Internet searches;
- Mathematics of relational databases;
- Ideas of linear algebra applied to finding single-error-correcting codes;
- Number theory used in the technical details of RSA public-key cryptography;
- Cryptography protocols for electronic communication, such as digital signatures and secure email;
- General understanding and examples of the P vs. NP problem;
- Use of vertex-edge graphs in constructing a Huffman data compression code;
- Data compression ideas, such as prefix-free codes, and lossy and lossless algorithms.

Thus, in different ways for different students, the mathematics of information processing and the Internet is essential mathematics to be included in a 21st Century high school curriculum.
References


