Coding Theory on the Towers of Hanoi

Ingrid Nelson
The College of William and Mary
ilnels@mail.wm.edu

August 11, 1995

Abstract

For a ternary code, distance is redefined based on the Towers of Hanoi puzzle. It is shown that, using Hanoi distance, a perfect one error-correcting code can be found for any length n ternary code. A formula for finding the number of codewords is given, and the codewords are defined. As well, a decoding algorithm is given and proved.

Keywords: Error-Correcting Codes, The Towers of Hanoi, Graph Theory

1 Introduction

When sending information, it is useful to convert messages into strings of numbers, which will then be received and decoded as the original messages. However, due to the imperfection inherent in manmade channels, some strings will be received incorrectly and must be decoded to discover which message was sent. In coding theory, the idea of the distance between two strings of symbols, called words or vectors, is used for such decoding. Typically, the Hamming distance (Hill, p. 5) is used for decoding, as it corresponds to a common channel used to transmit messages. However, a code based on the Hamming distance does not guarantee that any received vector can be unambiguously decoded. I found that when distance is defined differently, as the number

of moves it takes to get from one configuration to another in the Towers of Hanoi puzzle, an unambiguous, one-error-correcting code can be found for any given word length. Although this may not correspond to any currently used transmitting channels, it is interesting to see how such a convenient code can be defined.[1]

2 Previous Work

All definitions in this section are taken from Hill.

In basic coding theory, a code is defined as a q-ary (n, M, d)-code. Here, q is the cardinality of the alphabet over which codewords will be written. For example, the alphabet for a binary code is {0, 1}, so q = 2. The other parameters of the code are the length n of the string of symbols, called a word or vector, the number of codewords M, and the minimum distance d between codewords. Thus, if a code C consisted of codewords {000, 111}, it would be a binary (3, 2, 3)-code. This is using the Hamming distance, which measures the number of positions in which two codewords differ.

If a message is received which is not one of the codewords, errors are corrected using "nearest neighbor decoding"; that is, a word is decoded as the codeword closest to it by the Hamming distance. Naturally, it is desirable to be able to decode any possible word—any length n vector chosen from the q-symbol alphabet—unambiguously. A code in which all words can be decoded as exactly one codeword is called a perfect code.

With Hamming distance, non-trivial q-ary perfect codes exist for lengths n that satisfy:

$$n = \frac{q^r - 1}{q - 1}$$

for any integer r and prime power q. Perfect codes also exist for some other parameters specified by Hamming and Golay. Thus, perfect codes can only be found for certain lengths even when an alphabet with prime power size is used. Ideally, of course, we would like to be able to find perfect codes for any set of parameters. But barring that,

[1]This research was conducted during an NSF Research Experience for Undergraduates program at Oregon State University. I would like to thank Paul Cull, OSU professor of Computer Science, for providing direction and consultation.

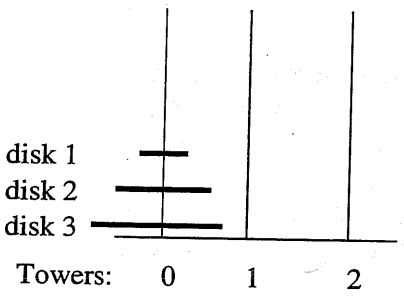
it would be useful if fixing some parameter, for example q, led to the existence of a perfect code for any length n.

Essentially, the Towers of Hanoi code, in which q = 3, yields this family of perfect codes.

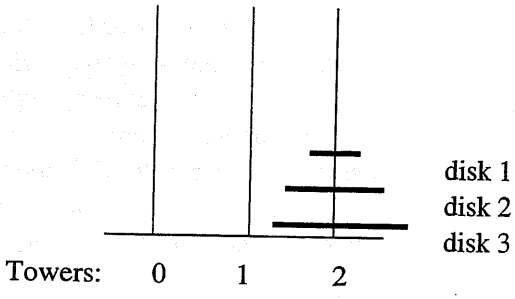
3 Definitions

Definitions in this section are similar to those in Hinz.

The Towers of Hanoi puzzle consists of three towers and n disks. Initially, all the disks are stacked, from largest to smallest, on the first tower. The object of the game is to move all disks onto the third tower so that they are stacked from largest to smallest. However, only one disk can be moved in a turn, and at no time can any larger disk rest on top of a smaller disk. We can name the towers 0, 1, 2, and we can name the disks 1 through n, with 1 being the smallest disk.

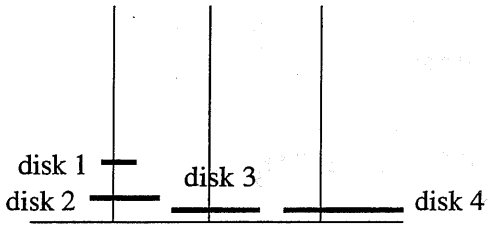


Initial configuration: 3 disks



Final configuration: 3 disks

A code based on the Towers of Hanoi puzzle redefines the distance between words in a ternary code. Consider a length n vector over the set {0, 1, 2}. This can be said to represent the positions of n disks on a three-towered puzzle. Thus, the word 0012 would correspond to the position:



The ith position in the word corresponds to the ith disk in the puzzle, and the symbol in this position is the tower this disk is on. Notice, each vector corresponds to exactly one configuration, because smaller disks must rest on larger disks. Also, any word will represent a Towers of Hanoi configuration, so a bijection exists between all ternary words of length n and all Towers of Hanoi configurations.

Then, we can define the Hanoi distance between vectors $\mathbf{x}$ and $\mathbf{y}$, written $d_{ToH}(\mathbf{x}, \mathbf{y})$, as the minimum number of moves it takes to get from $\mathbf{x}$ to $\mathbf{y}$ according to the rules of the Towers of Hanoi.

Thus, $d_{ToH}(000, 100) = 1$, but $d_{ToH}(000, 001) = 7$. This shows that a Hamming distance of one between vectors is a necessary condition for a Hanoi distance of one, but the converse does not hold.

If we assume $\{x, y, z\} = \{0, 1, 2\}$, and let any word have the symbol x in the first k positions, with $k \geq 1$, we can state the rules for making one move on the Towers of Hanoi in the following form:

1. For some word $\mathbf{w} = w_1....w_n$, w_1 can be changed to y or z.

2. If $w_{k+1} = y$, it can be changed to z. Like wise, a z in this position can be changed to y.

These are the only changes that can be made in one move.

This generalization is helpful in understanding Hanoi distance, and will be used in some upcoming proofs.

The Hanoi distance satisfies the crucial properties of distance:

1. $d_{ToH}(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$.

2. $d_{ToH}(\mathbf{x}, \mathbf{y}) = d_{ToH}(\mathbf{y}, \mathbf{x})$.

3. $d_{ToH}(\mathbf{x}, \mathbf{z}) \leq d_{ToH}(\mathbf{x}, \mathbf{y}) + d_{ToH}(\mathbf{y}, \mathbf{z})$.

For any Hanoi code, then, a one-error correcting code is one in which the minimum distance between codewords is three. This follows from a theorem which states that if a code's distance is equal to $2t + 1$, it corrects t errors (Hill, p. 7.)

Since all Hanoi codes are ternary, my problem was to find all perfect one-error correcting codes of this form. As well, I hoped to find a

formula for the number of codewords in such a code, a way to generate them, and a reasonable decoding algorithm.

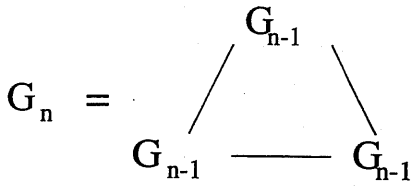
4 The Structure of the Hanoi Code

This structure is also defined in Hinz, although I did not come across his paper until I had finished my proofs, which will be given here in full.

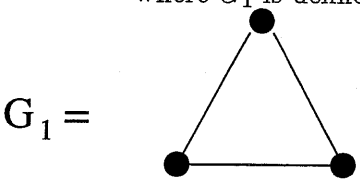
This section will construct a graph of ternary n-space in which the distance between vertices corresponds to the Towers of Hanoi distance between vectors. I will also present a way of choosing codewords on this graph which yields a perfect, one-error correcting code for any length n.

4.1 The Hanoi Graph

Consider a recursive graph G_n such that

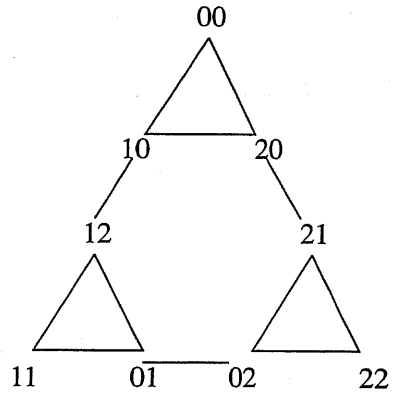


where G_1 is defined:



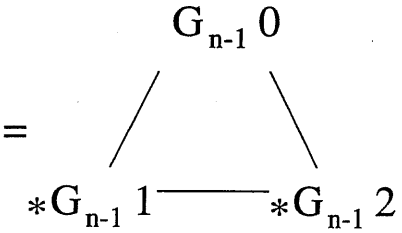
In these figures, a filled in circle represents a vertex, and a line represents a two-way edge between vertices. Clearly, such a graph will have 3^n vertices, so it has one vertex for each ternary word of length n. When these words are associated with the graph, we allow the bottom two G_{n-1}'s to be rotations of the top G_{n-1}. This preserves the symmetry of the graph, but allows the repetition vectors 0...0, 1...1 and 2...2 to be in the corners. For example, G_2:

$G_2 =$



In fact, if $G_n x$, where $x \in \{0, 1, 2\}$, is defined to be the graph G_n in which all words have x in the n+1th position, and $^*G_n$ to be a rotated graph of G_n, we can define:

$G_n =$

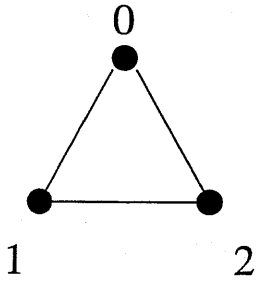


Now, I wish to show that a graph of this form is representative of the Hanoi distance between words.

Theorem 1 *The ternary words of length n can be arranged on the graph G_n given above such that the Hanoi distance between words is represented by the number of edges between the vertices associated with those words.*

Proof: Associate the length 1 ternary words with the graph G_1 as follows:

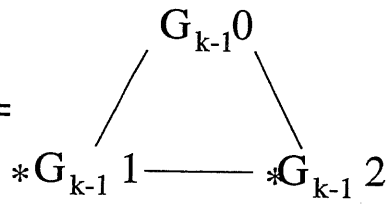
$G_1 =$



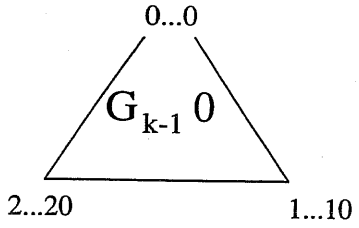
Clearly, the Hanoi distance between distinct vertices is one, and this is reflected by the edges of the graph.

Now, suppose the theorem is true for G_{k-1}. Then,

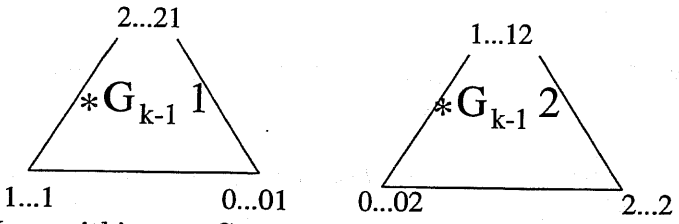
G_k =



But, since G_k must be arranged with the repetition vectors in the corners, we know we have:



G_k =



Now, within any $G_{k-1}x$ or its rotation, the edges must represent Hanoi distance by the induction hypothesis. So, it remains to show

d_{ToH} (2..20, 2...21) = 1,

d_{ToH} (1...10, 1...12) = 1 and

d_{ToH} (0...01, 0...02) = 1.

This is trivial. Thus, the graph of G_k takes the desired form and the theorem is true by induction. □

5 A Perfect Hanoi Code

So far, a code has been defined which, like most codes, uses vectors in n-space as its codewords, but unlike most codes, defines distance based

on the Towers of Hanoi puzzle. As well, this code has been related to a graph structure. However, the real importance of this code is not in its unusual definition of distance, but in its elegant properties. In this section, I will show that a perfect, one-error correcting code exists for any Hanoi code of length n. It turns out that the Hanoi codes have a slightly different general structure for odd and even lengths n, but both cases can be represented using two models.

5.1 The Parity Lemma

One of the main differences between the choice of codewords in the odd and even cases is the number of repetition vectors which are codewords. On the graph of G_n, these are the three corner vertices.

First, a definition is necessary.

Definition A *sphere* of radius s in the Towers of Hanoi code consists of a word **x** and all words at Hanoi distance s from **x**. (Hill, p. 18). For such a code, a sphere of radius one will contain 3 vectors if **x** is a corner vertex of the graph, and 4 vectors if **x** is not a corner vertex.

Proof. The corner vertices of G_n are the repetition vectors, as shown above. Since, in this vector, all the disks are stacked on one tower, only the top disk can be moved. And, there are only two free towers, so it can only be moved two different places. Thus, there are two vectors of Hanoi distance one from a corner vertex, and therefore 3 vectors in the radius one sphere.

However, any non-corner word will be Hanoi distance 1 from exactly three other words. This follows from the rules given for distance one moves in section three. □

In an s-error correcting code, all words within the radius s sphere of the codeword **x** are decoded as **x**(Hill, p. 19.) The Parity Lemma rests on this fact.

Lemma 1 *If a perfect, one-error correcting Hanoi code exists, then*

1. *For even n, all three corners of G_n are codewords;*
2. *For odd n, exactly one corner is a codeword.*

Proof. In a perfect code, every vector decodes unambiguously as a codeword. Thus, the spheres must be disjoint and cover the entire space. There are 3^n distinct words in ternary n-space, so if there are j

corner codewords and k non-corner codewords, we need to show that $3j + 4k = 3^n$.

Case 1. Let n = 2r. Suppose all three corners are codewords. Then the space is reduced to 3^{2r} - 9 words. If this can be divided evenly into four, there are spheres of non-corner codewords which exactly cover the remaining space. And indeed, 3^{2r} - 9 $\equiv$ 1-1 = 0 [mod 4].

However, if 0, 1, or 2 corner codewords are used, it is easily seen that

3^{2r},

3^{2r} - 3 and

3^{2r} - 6

are not evenly divisible by four. Thus three corner codewords must be used.

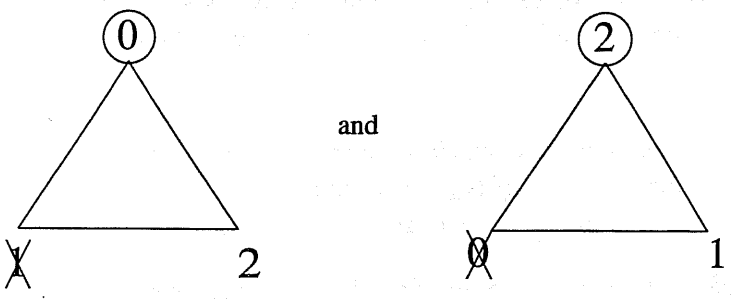
Case 2. Suppose n = 2r + 1. We claim that only one corner can be used. The argument is as above:

3^{2r+1} - 3 = 3^{2r} * 3 - 3 $\equiv$ 1*3 - 3 = 0 [mod 4].

However, the equivalence does not apply for any other number of corner codewords. □

5.2 Choosing Codewords on G_n

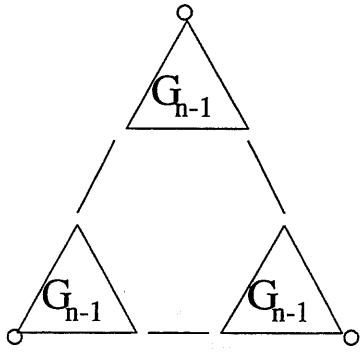
In this section, I will define a method of choosing codewords on G_n, respective of the parity of n, and prove that such a choice yields a perfect one-error correcting code. In this section, the graph G_n or U_n will represent a graph in the shape of some G_n on which certain vertices are chosen as codewords. The choices are positional, not numeric. A hollow circle around a vertex specifies that it is a codeword; an X specifies that it is not. So, for example,



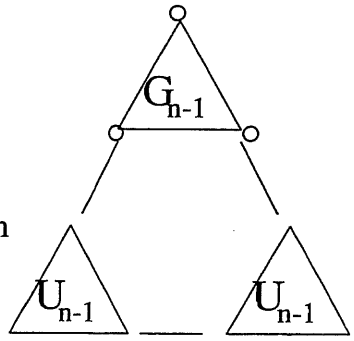
are the same graph, in which the lower right corners may or may not be codewords.

For the Hanoi code, the graphs corresponding to codeword positions are recursive, and defined as follows:

$G_n =$ For even n

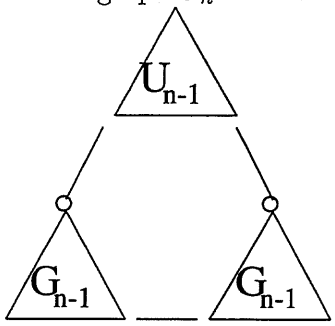


$G_n =$ For odd n

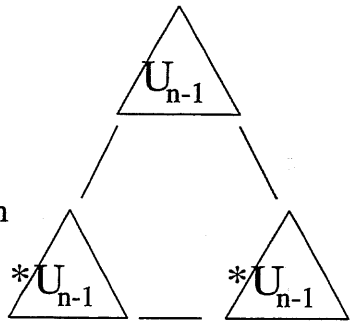


where the graph U_n is defined:

$U_n =$ For even n



$U_n =$ For odd n



As well, we define

$G_0 =$ o (a codeword) and $U_0 =$ X (not a codeword)

To show that these graphs create perfect, one-error-correcting codes, a lemma about the properties of the graph U_n is necessary.

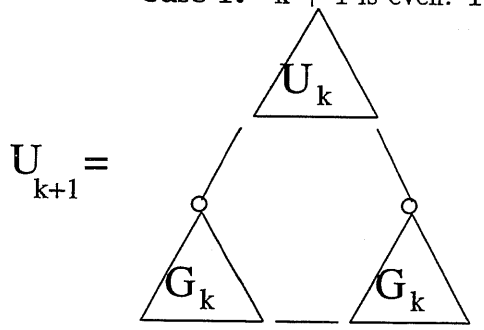
Lemma 2 *In the above graphs U_n,*

1. If n is odd, no word at Hanoi distance of one or zero from a corner vertex can be a codeword.

2. If n is even, no corner vertex is a codeword. As well, no word at Hanoi distance one from the top corner (in standard orientation) is a codeword.

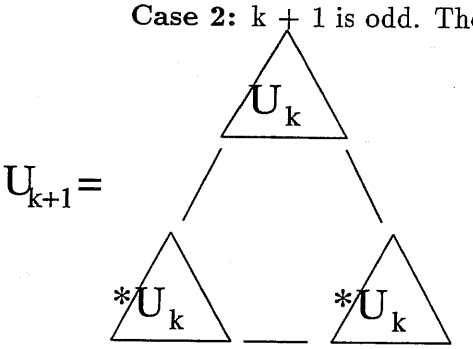
Proof. When n = 0 or n = 1, the given properties clearly apply. Now, consider n = k, and assume the hypotheses hold true for this k. Consider k + 1.

Case 1: k + 1 is even. Then



Since k is odd, by the induction hypothesis no word at distance one or zero from a corner vertex can be a codeword. Thus, this property is true of the top corner U_{k+1}.

Case 2: k + 1 is odd. Then



By the induction hypothesis, the top corner and the words at distance one from it are not codewords. Now, we want to show this is also true of the bottom two corners of U_{k+1}. But, the rotation of the bottom U_{k+1}'s has not been strictly defined, we can define it so that the corners of U_k with the desired property are the corners of U_{k+1}. □

5.3 Proving the Existence of the Perfect, One-Error Correcting Hanoi Code

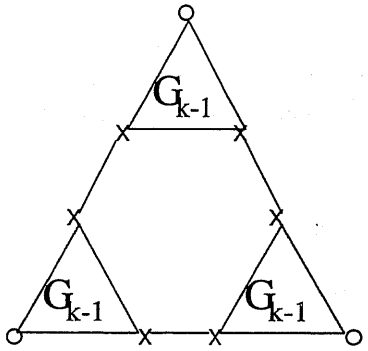
Now, we can show that G_n constructs a perfect code.

Theorem 2 *The above construction of the graph G_n gives a perfect, one-error correcting code for any n.*

Proof. For G_0 and G_1, it is trivial to show these codes are perfect and one-error correcting. Now, assume that for some n = k, the graph G_k yields a perfect, one-error correcting code. Consider n = k + 1.

Case 1: k + 1 is even. Since each G_k can have only one corner vertex as a codeword, we know

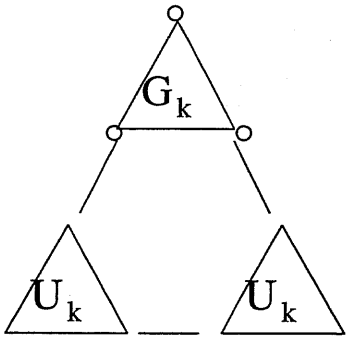
G_k =



Thus, since the corners of the G_k's which are distance one from each other cannot be codewords, any word in G_{k+1} will decode within G_{k+1}, which we know is perfect and one-error-correcting.

Case 2. k + 1 is odd. Then,

G_{k+1}=



G_k is perfect and one-error correcting. It remains to show that there is no ambiguity in decoding the top corners of the U_k's, and that all other words in U_k decode to exactly one codeword.

By the lemma in section 5.2, the top corner of U_k and the words at distance one from that corner cannot be codewords. Thus, the top corners of U_k can be unambiguously decoded as the bottom corners of G_k. The proof that all other words in U_k are at distance one from exactly one codeword is simple induction and will be omitted here. □

Thus, the Hanoi code has the useful property that a perfect, one-error correcting code exists for any word length n.

6 Features of the Hanoi Code

In the previous section, the existence of a perfect Hanoi code was shown using recursive graph structures. In this section, I will present some algorithms and formulas that can be used to generate and decode a Hanoi code.

6.1 The Number of Codewords

By examining several perfect Hanoi codes, I determined experimentally that a recursive formula for the number of codewords is

$$M(n) = 3(M(n-1))$$

for even n and

$$M(n) = 3(M(n-1)) - 2$$

for odd n. When I converted this to a non-recursive formula, I found

$$M(n) = 9^r - 6(\sum_{i=1}^{r} 9^{r-i}) = \frac{3^n + 3}{4}$$

and

$$M(n) = 9^r - 2(\sum_{i=1}^{r} 9^{r-i}) = \frac{3^n + 1}{4}$$

for n = 2r and n = 2r + 1, respectively.

Theorem 3 *The above formulas for M(n) represent the number of codewords in a perfect, one-error correcting Hanoi code, respective of parity.*

Proof 1. We know that there are 3^n ternary words of length n. We also know that the radius one sphere of a corner codeword contains three words, and that of a non-corner codeword contains four. In a perfect code, each word in the space must be in exactly one sphere. Thus, if k is the number of corner codewords, we need to show 3k + 4(M(n)-k) = 3^n.

Case 1. Consider the formula

$$M(n) = 9^r - 6(\sum_{i=1}^{r} 9^{r-i})$$

which applies when n = 2r. Let r = 1. Now, we know that if we want a perfect code of even length, three corners must be codewords. So, the space must be covered by 9 + 4 * (the number of non-corner codewords) = 9 + 4(M(2)-3). Since M(2) = 3, we have 9 - 0, which covers the nine words in ternary 2-space.

Suppose n = 2k, and assume the hypothesis is true. Then,

$$4(9^k - 6(\sum_{i=1}^{k} 9^{k-i}) - 3) + 9 = 3^{2k}$$

This implies

$$4(9^k - 6(\sum_{i=1}^{k} 9^{k-i}) = 3^{2k} + 3$$

Consider n = 2(k + 1). Then,

$$4(9^{k+1} - 6(\sum_{i=1}^{k+1} 9^{k+1-i} - 3) + 9 = 4 * 9(9^k - 6(\sum_{i=1}^{k} 9^{k-i}) - 27 = 9(3^{2k} + 3) - 27 = 3^{2(k+1)}$$

Case 2. If n = 2r + 1,

$$M(n) = 9^r - 2(\sum_{i=1}^{r} 9^{r-i})$$

Consider r = 1. We know we must have exactly one corner codeword for the code to be perfect, so we have 4(9 - 2(1) - 1) + 3 = 27 = 3^3.

Suppose the assumption is true for some r = k. Then

$$4(9^k - 2(\sum_{i=1}^{k} 9^{k-i} - 1) + 3 = 3^{2k+1}$$

which implies

$$4(9^k - 2(\sum_{i=1}^{k} 9^{k-i}) = 3^{2k+1} + 1$$

Thus, when we consider r = k+1, we have

$$4(9^{k+1} - 2(\sum_{i=1}^{k+1} 9^{k+1-i} - 1) + 3 = 3^{2(k+1)+1}$$

In the above proof, the cases of n = 0 and n = 1 were not dealt with, but the values of M(0) and M(1) are easily determined to both be 1 by the graph structure. □

After I did this proof, another, simpler proof occurred to me:

Proof 2. Unless a codeword is a corner vertex, there are four words which will be decoded as that word. Thus, if we add one more word for each corner codeword to the total number of words in the space and divide this sum by four, we should have the number of codewords. And indeed, these quotients come out to be the simpler fractions given above. □

6.2 Generating Codewords

By observation, I discovered a property of the codewords of the Hanoi code which may facilitate the construction of a generating algorithm. The following lemma is needed to prove this property.

Lemma 3 *Let $[0_v]$ denote the number of 0's in the word* **v**. *Suppose $[0_v]$, $[1_v]$ and $[2_v]$ are all even numbers when the length of* **v** *is even, and that $[1_v]$ and $[2_v]$ are even and $[0_v]$ is odd when n is odd. Then, if* **v** *and* **w** *are words of this form for some n, d_{ToH} (***v***, ***w***) $\geq$ 3.*

Proof. For some fixed n, suppose there are two distinct words of the above form, **v** and **w**, which have Hanoi distance less than or equal to three.

However, the words cannot be at Hanoi distance one from each other, because this would imply a Hamming distance of one between them. And, changing only one position in such a word would disturb the parity requirement.

So it must be the case that d_{ToH} (**v**, **w**) = 2. This implies that both words are at Hanoi distance one, and therefore at Hamming distance one, from some word **u**. Assume that the symbol x $\in$ {0, 1, 2} occupies the first k places of **v** and the first j places of **w**. Then, by the numeric rules for one move on the Towers of Hanoi, we know one of the following three cases applies:

1. $\mathbf{u} = yv_2...v_n = yw_2...w_n$,
2. $\mathbf{u} = zv_2...v_n = zw_2...w_n$, or
3. $\mathbf{u} = v_1...v_i'...v_n = w_1...w_j'...w_n$ where $v_i' \neq v_i$ and $w_j' \neq w_j$

The first two cases imply that $v_i = w_i$ for all i $\geq$ 2. However, if this were true, it would follow that d_{ToH} (**v**, **w**) = 1, which is a contradiction.

Now, consider case three. If i = j, again it follows that d_{ToH} (**v**, **w**) = 1. Suppose i < j. Now, we know that $v_i \neq$ x and $w_j \neq$ x by the rules that define Hanoi distance one. But if i < j, we know $w_i =$ x. Since v_i can only be changed to y or z, the two words in case three cannot be the same.

Thus the lemma is true by contradiction. □

Now, it remains to show that the number of words of this form is equal to the number of codewords in a perfect Hanoi code.

Lemma 4 *When n is even, there are* $\frac{3^n+3}{4}$ *words where [0], [1] and [2] are even. When n is odd, there are* $\frac{3^n+1}{4}$ *words where [1] and [2] are even and [0] is odd.*

Proof. Consider the trinomial expansion of $(x + y + z)^n$. If we assume {x, y, z} = {0, 1, 2}, the coefficient of $x^i\ y^j\ z^k$ will be the number of words of length n containing x in i positions, y in j positions, and z in k positions. Thus, we can count codewords with a certain number of zeros, ones and twos using this formula.

Now, define x_e to be the number of terms in the expansion in which x has an even exponent, x_o to be the number of words in which x has an odd exponent. As well, define x_e & y_e to be the number of words in which the exponents of both x and y are even. We can group terms according to the parity of the exponents of x and y, since this forces a z to have a specific parity.

Then, when x=y=z=1, we get all 3^n words in the space, and we can group them as follows:

$(1+1+1)^n = 3^n = (x_e\ \&\ y_e + x_e\ \&\ y_o + y_e\ \&\ x_o + y_o\ \&\ x_o)$.

As well, if we consider cases in which x and/or y is equal to -1, we have

$(-1+1+1)^n = 1^n = (x_e\ \&\ y_e + x_e\ \&\ y_o - y_e\ \&\ x_o - y_o\ \&\ x_o)$

$(1-1+1)^n = 1^n = (x_e\ \&\ y_e - x_e\ \&\ y_o + y_e\ \&\ x_o - y_o\ \&\ x_o)$

$(-1-1+1)^n = -1^n = (x_e\ \&\ y_e - x_e\ \&\ y_o - y_e\ \&\ x_o + y_o\ \&\ x_o)$

Now, if we add these four expressions, all terms cancel except those in which both x and y have even exponents. Thus, if n is even we have $3^n + 3 = 4(x_e \ \& \ y_e)$, which implies $x_e \ \& \ y_e = \frac{3^n+3}{4}$. If n is odd, $3^n + 1 = 4(x_e \ \& \ y_e)$, so $x_e \ \& \ y_e = \frac{3^n+1}{4}$. In this case, we can assume z represents the symbol 0, which appears an odd number of times in a word in which [x] and [y] are even. □

Thus, there are exactly the same number of codewords as there are words of the specified form.

The result below follows naturally from the above lemmas.

Theorem 4 *If the length of a perfect Hanoi code is odd, the codewords are exactly those words in which [1] and [2] are even, and [0] is odd. If the length is even, the codewords are exactly those words for which [0], [1] and [2] are even.*

Proof. In a perfect, one-error correcting code any codeword has distance exactly three from the nearest codewords. Since the code is perfect, if there exists a set of words with the same cardinality as the set of codewords, these words must be distance three or less from each other. But, we've shown that no word with the desired construction can have a distance less than three from any other word of this kind. Thus, any such word must have a distance of exactly three from the closest codewords. And it follows that t hey constitute the codewords of the perfect code. □

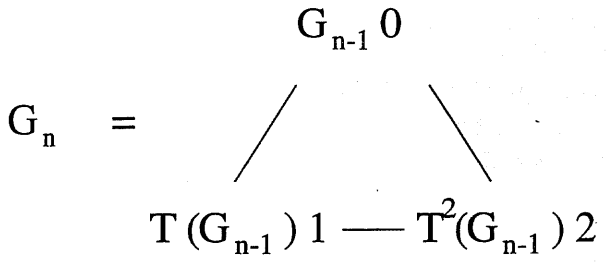
Because of this property, the Hanoi code is cyclic; that is, any cyclic rotation of a codeword is also a codeword. I have not written an algorithm to generate these codewords, but this rule should facilitate the creation of one.

6.3 A Decoding Algorithm

Due to the recursive nature of the Towers of Hanoi graphs, it seems natural to define a recursive decoding algorithm which looks at the last digit of the received word, decides whether to keep or change it, and the passes the truncated word to another decoding routine. It also seems natural to have four decoding routines based on the graphs: two routines each for U_n and G_n depending on the parity of n.

However, because some of the graphs are rotated, an adjustment will have to be made to the word when it is passed between routines.

If the graph G_n is oriented so the vector **2** is in the lower right corner, we can define:



where $T^k(G_n)$ is the permutation $\overleftarrow{0 \to 2 \to 1}$ applied k times to the words of G_n. It can easily be seen that this is the same as the rotated G_n which has been used throughout the paper.

In the decoding algorithm, we first want to determine whether the received word $\mathbf{y} = y_1 ... y_n$ is or is not a codeword. So, read in n and determine its parity. Then, define counters [0], [1] and [2], run through the word and imncrement the value of the counter [x] every time the symbol x is encountered. When this is done, check the parities of the counters. If they match the rule given in section 6.2, the word is a codeword and can be returned unchanged.

If the word is not a codeword, begin the decoding algorithm. The following four routines are used. The variable T is a member of the set {0, 1, 2}, and represents the number of permutations which should be performed on the word before decoding begins. The function ApplyT, which will not be defined here, performs this task and returns the permuted **y**. The variable **y'** will be used to store the decoded word.

Routine 1: Decodes within G_n for even n.
Procedure DecodeGEven (n, **y**, **y'**)
store y_n in the nth position of **y'**
set y_n = T; remove y_n from **y** and set n = n-1
call ApplyT(**y**, T); returns the permuted **y**
call DecodeGOdd(n, **y**, **y'**); returns decoded **y'**
call ApplyT(**y'**, 3-T)
return **y'**
Routine 2: Decodes within G_n for odd n.
Procedure DecodeGOdd (n, **y**, **y'**)
if n=1, **y'** = 0; return y'.

if $y_n = 0$, store 0 in y'_n, call DecodeGEven
when **y'** is returned, call ApplyT(**y'**, 3-T)
else if $y_1{=}y_2{=}...{=}y_{n-1}$ and $y_n \neq 0$,
store 0 in y'_n; return **y'**
when **y'** is returned, call ApplyT(**y'**, 3-T)
else store y_n in y'_n, set T=y_n, truncate **y**
call ApplyT(**y**, 2T), call DecodeUEven
call ApplyT(**y'**, 2T)
return **y'**

Routine 3: Decodes within U_n for even n.

Procedure DecodeUEven (n, **y**, **y'**)
if $y_n = 0$ and $y_1{=}...{=}y_{n-1}$
if $y_{n-1}{=}1$, let $y'_{n-1} = 2$ and vice versa
return **y'**
else if $y_n = 0$ and Not($y_1{=}...{=}y_n$)
set T=y_n, truncate **y**
call ApplyT(**y**, y_n), call DecodeUOdd
else if $y_n \neq 0$, call DecodeGOdd
return **y'**

Routine 4: Decodes within U_n for odd n.

Procedure DecodeUOdd (n, **y**, **y'**)
set T=y_n; store y_n as y'_n
call ApplyT (**y**, T)
call DecodeUEven; returns **y'**
call ApplyT (**y'**, 3-T)
return **y'**

This decoding algorithm follows directly from the structures of G_n and U_n.

7 Conclusion

A perfect, one-error-correcting code exists for any length n ternary code in which distance is defined according to the Towers of Hanoi puzzle. If n is even, there are $\frac{3^n+3}{4}$ codewords, which are exactly those words in ternary n space containing an even number of zeros, ones and twos. If n is odd, there are $\frac{3^n+1}{4}$ codewords, which are exactly those words containing an even number of ones and twos, and an odd

number of zeros. The codes can be represented on recursive graphs, from which the decoding algorithm is derived.

A few refinements could be made on the perfect, one-error correcting Hanoi code. For example, the algorithm to generate codewords has not been written, and a more elegant decoding algorithm may exist.

As well, other mathematicians may be interested in finding perfect Hanoi codes which correct more than one error, or perfect error-detecting Hanoi codes. Clearly, the repetition Hanoi code on any length n corrects n-1 errors perfectly, but nothing is known about values between 1 and n-1.

Finally, like so many beautiful things, the Hanoi code has no practical use of which I am aware. Because of the code's elegant properties, it would be worthwhile to look for an application of this material.

8 Bibliography

1. R. Hill. *A First Course in Coding Theory*, Oxford University Press, Oxford, 1986.

2. A. M. Hinz. Pascal's Triangle and the Tower of Hanoi, *American Mathematical Monthly*, June-July 1992, pp. 538-543.