Coding Theory on the Generalized Towers of Hanoi

Danielle Arett

August 1999

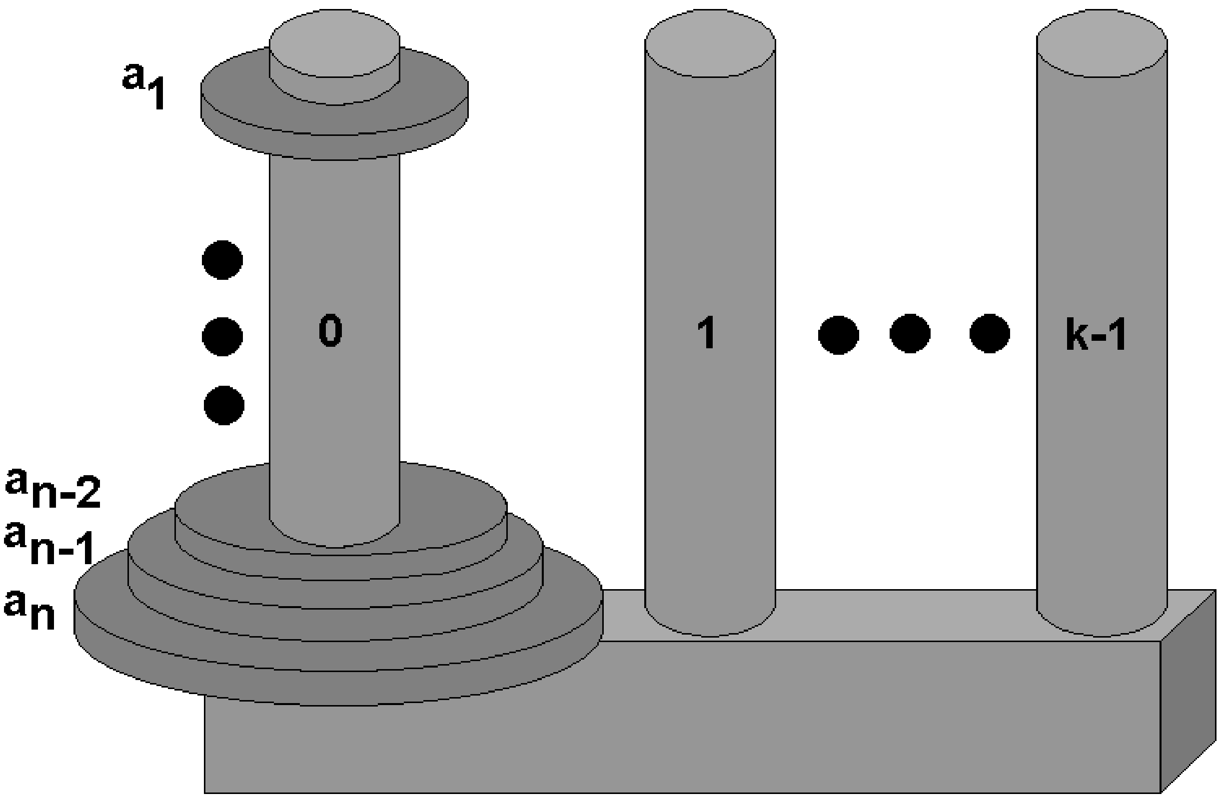


Figure 1

Coding Theory on the Generalized Towers of Hanoi

Danielle Arett
Augsburg College
Minneapolis, MN
arettd@augsburg.edu

Advisor: Professor Paul Cull
Department of Computer Science
Oregon State University
Corvallis, Oregon
pc@cs.orst.edu

August 12, 1999

Abstract:

An attempt is made to extend the coding theory based on the Towers of Hanoi puzzle to the generalized Towers of Hanoi with more than three pegs. A three-dimensional graph is created for the case of four pegs, and a recursive construction for this graph is given based on the number of disks used. Proofs that no perfect one error-correcting code (P1ECC) or P2ECC exist on the graph for four pegs with three desks are given.

Introduction

When information is sent electronically, messages are often converted into strings of numbers. When received, these strings are decoded as the original message. Unfortunately, due to both human folly and machine inaccuracy, the strings may contain errors, so error-correcting codes are constructed to enable us to obtain the original message. These error-correcting codes may be studied as graphs consisting of vertices and deges. The strings are each assigned to a vertex, and an edge between two vertices represents a distance of 1 between the two strings assigned to those vertices. Depending on how distance is defined, one may study graphs to see if they contain an error-correcting code.

1.0 Towers of Hanoi

The Towers of Hanoi puzzle has been of mathematical interest for decades. It consists of 3 pegs and a number of different sized disks which are initially placed on the first peg in order of size, the smallest on top. The object of the puzzle is to stack all the disks on the third peg by only moving one disk at a time and placing disks only on larger disks. By labeling the disks and pegs, one may create a perfect one error-correcting code whose words are base 3 and whose distance between words is defined as the minimum number of legal moves between configurations on the puzzle (1).

One variant of the Towers of Hanoi puzzle is the generalized Towers of Hanoi which has more than three pegs. It has been studied by computer scientists in connection with material handling and production scheduling (Hinz 133), and recursive solutions to the general puzzle have been found. However, I have found no documentation of relating the error-correcting codes of the original Towers of Hanoi puzzle with the generalized puzzle. In order to do this, some simple definitions and notation are necessary.

1.1 Definitions & Notation

See Figure 1. In a generalized Towers of Hanoi puzzle and graph, we let...

- k = # of towers, $k \geq 3$
- Towers are named 0, 1, 2, ..., k-1
- n = # of disks, $n \geq 1$
- Disks are named $a_1,a_2,...,a_n$
- TH(k,n) is a generalized Towers of Hanoi puzzle with k towers and n disks

$A = a_n...a_2a_1$ is a *word* describing a specific configuration on the puzzle where a_i is defined by the tower on which the *i*th disk rests

- The *distance* from one word A to another word B is the minimum number of legal moves it takes to get from the configuration of A to the configuration of B on the generalized Towers of Hanoi puzzle. We write D(A,B) = j for some $j > 0 \in \mathbb{Z}$ to denote distance between A and B, and we write Dj(A) = $\{B : D(A,B) = j\}$.

  Example $D(012,020) = 2$ *and* $D2(12) = \{000, 001, 002, 010, 020, 011, 211\}$.
- A word B is *covered* by a codeword A on a *d*ECC if $D(A,B) \leq d$.

2.0 Reve's Puzzle — TH(4,n)

The generalized Towers of Hanoi with four pegs, or Reve's puzzle, has been studied in both computer science in relation to computer programming and algorithm complexity as well as in combinatorial mathematics (Lu & Dillon 3). The rules of the puzzle are the same as TH(3,n), and both recursive and iterative solutions have been found for the puzzle. We can construct a code similar to the TH(3,n) code based on Reve's Puzzle by creating words in base 4 rather than base 3. One favorable property of the TH(4,n) graph is that despite its complexity, it has a simple recursive construction as we increase n.

2.1 1 Disk → 2 Disks — TH(4,2)

The initial configuration on Reve's puzzle consists of all disks on peg 0, so the only possible action is moving the top disk to peg 1, 2, or 3. This is equivalent to having only one disk on the puzzle, and the action can be graphically described by a tetrahedron whose four vertices are words and whose edges represent a distance of 1 between the words (see Figure 2). If the puzzle has two disks, the graph expands to four tetrahedrons and sixteen vertices, with specific edges between them (see Figure 3). The process for choosing codewords is quite simple. First, choose an arbitrary word to be a codeword. Then, all words distance 1 from that codeword are covered by it. All other words distance 1 from each covered word may not be codewords, and they need to be covered by another codeword. We choose more codewords to cover these words, being sure that no two codewords are less than distance 3 apart. For instance, on the TH(4,2) graph, we may choose the word 10 as a codeword. Then, 20, 30, 11, 12, and 13 are covered by 10, and and 21, 22, 23, 31, 32, 33, 02, and 03 must be covered by other codewords. We could choose either 00 to cover 02 and 03, or we could choose 01 and cover 02, 03, 21, and 31, but 22, 23, 32, and 33 will not be covered. Since all words distance 1 from 22 and 33 are already distinguished as non-codewords, we will not have a P1ECC if 10 is a codeword. Figure 3 does contains a P1ECC with 00, 11, 22, and 33 as codewords. We see that the non-codewords on each tetrahedron are decoded to the codeword on that tetrahedron, and each codeword is distance 4 apart. Thus, no two codewords are adjacent, and each non-codeword is adjacent to exactly 1 codeword. The choice of codewords on this graph is unique since any other choice will not generate a P1ECC.

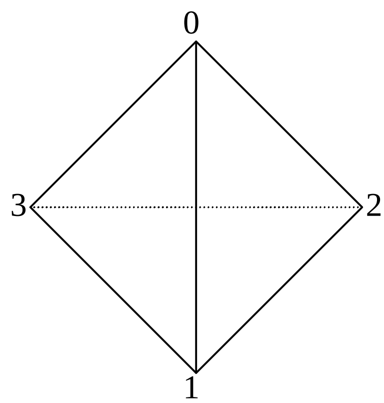


Figure 2

If we choose any one word as a codeword, then Figure 2 contains a P1ECC. The choice of the codeword is not unique, but each is homogeneous to the other.

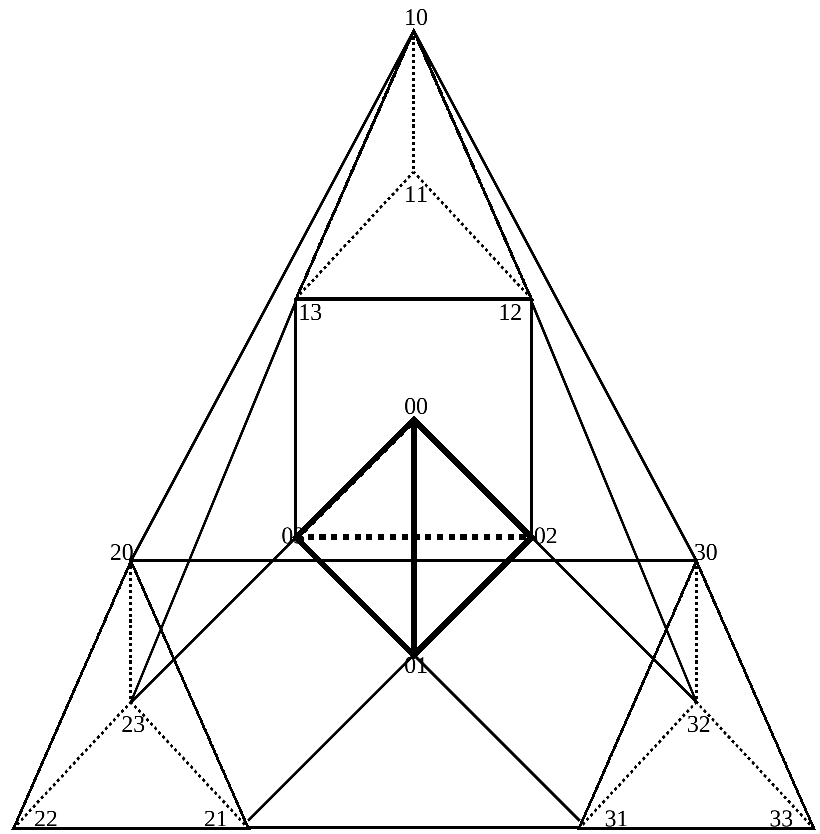


Figure 3

2.2 TH(4,3)

The graph of Reve's puzzle becomes surprisingly complex when we add a third disk. To understand the graph, different views may be neccessary. If we ignore the 3-dimensional quality of the tetrahedrons, we can get an idea of how the bases of the tetrahedrons connect (see Figure 4). Then, if we stretch the tops of the tetrahedrons, we can see patterns in the edges between them (see Figure 5).

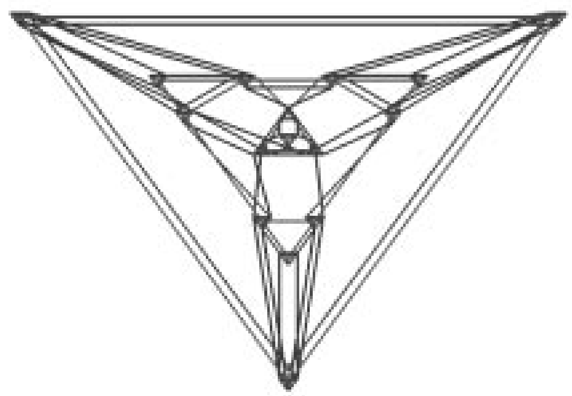


Figure 4

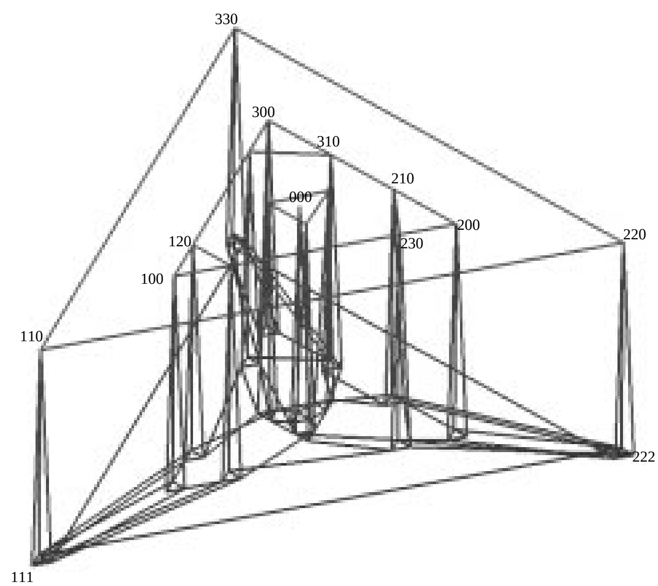
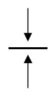
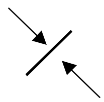


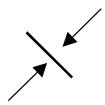
Figure 5

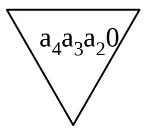
2.3 Recursive Construction

Notation G_n = *The graph of Reve's puzzle using n disks.*

 = *a horizontal reflection in the xy-plane*

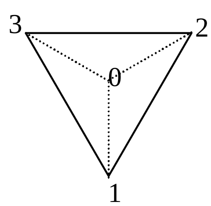
 = *a reflection across a line with slope* $\sqrt{3}$

 = *a reflection across a line with slope* $-\sqrt{3}$

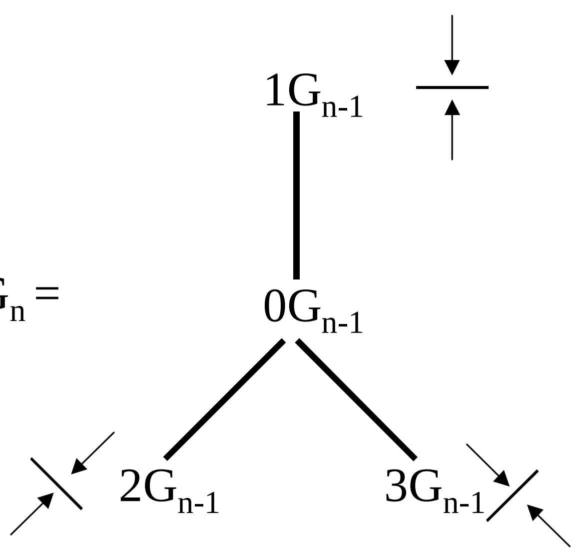


= *the tetrahedron containing the words* $a_4a_3a_20$, $a_4a_3a_21$, $a_4a_3a_22$, *and* $a_4a_3a_23$

$G_1 =$

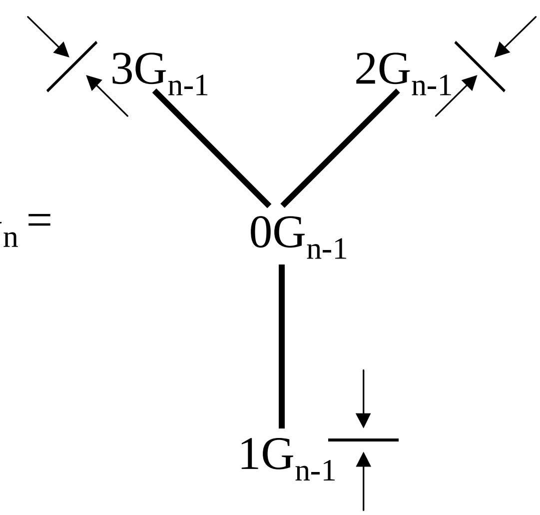


$G_n =$



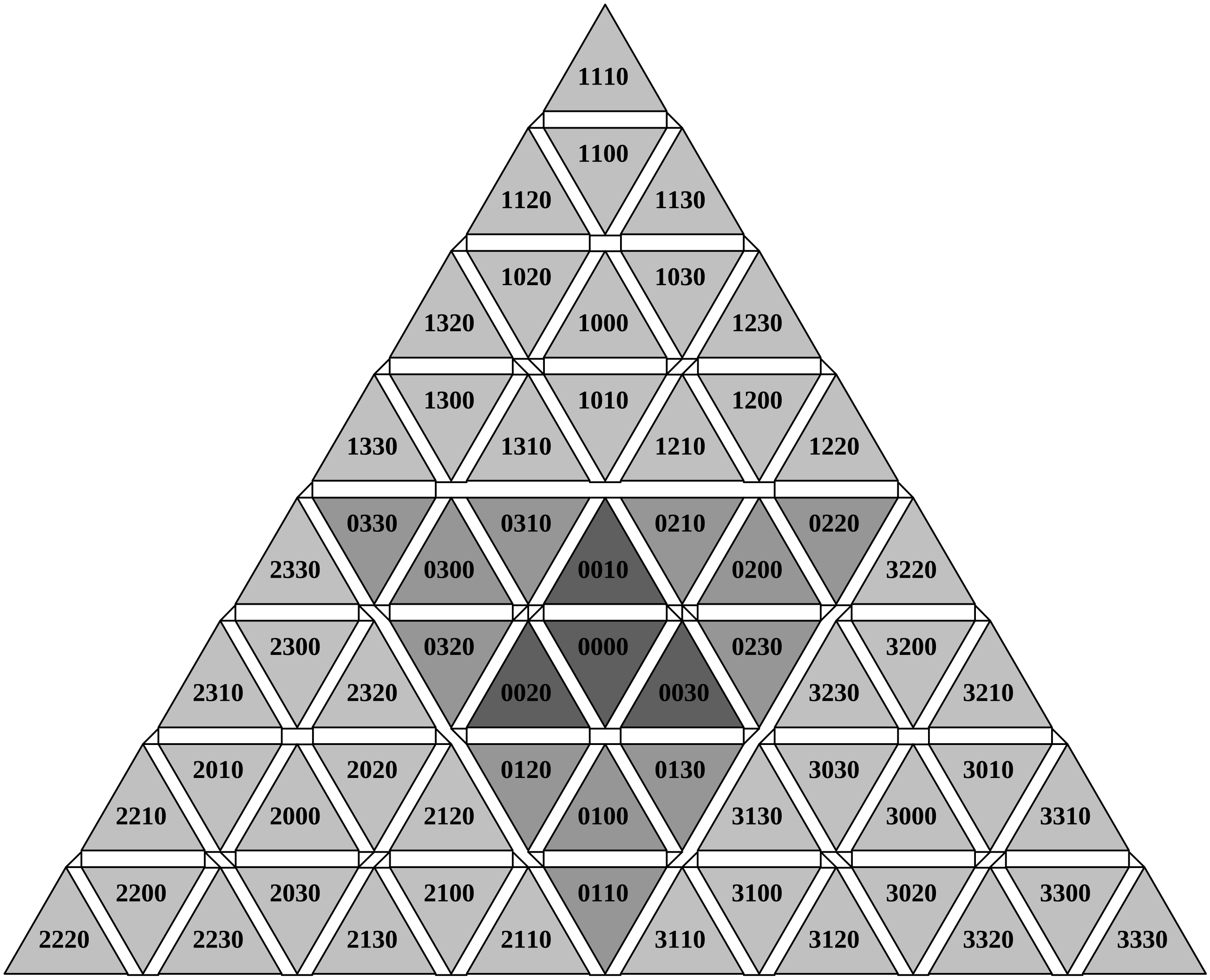
for even n

$G_n =$



for odd n

Example *The graph for Reve's puzzle with* 4 *disks seen from above is as follows*:



Example *Though all the edges are not shown on this graph, one may get a better understanding of the graph by looking at it this way*.

2.4 Codes on TH(4,3)

After investigating the structure of the TH(4,3) graph, we can see that no P1ECC exists for TH(4,3). First, it is neccessary to define more notation:

Notation $\Delta a_3a_2 0$ *denotes the tetrahedron containing the words* $a_3a_2 0$, $a_3a_2 1$, $a_3a_2 2$, *and* $a_3a_2 3$.

Example **Example** $\Delta 110$ *includes the words* 110, 111, 112, *and* 113.

2.41 P1ECC

Theorem (1) *For a P1ECC, the tetrahedrons* $\Delta 000$, $\Delta 110$, $\Delta 220$, *and* $\Delta 330$ *must contain codewords*.

Proof *For a P1ECC, all words are either codewords or share an edge with a codeword. Since* 000, 111, 222, *and* 333 *share edges only with words on their respective tetrahedrons, they must* *either be codewords, or their tetrahedrons must contain codewords.*

Theorem (2) 001, 002, 003, 110, 112, 113, 220, 221, 223, 330, 331, *and* 332 *cannot be codewords.*

Proof *In order to produce a contradiction, assume* 001 *is a codeword. Then* 000, 002, 003, 021, *and* 031 *are decoded to* 001. *Furthermore, all other words distance* 2 *from* 001 *cannot be codewords. Since* $D(001, 032) = 2$, 032 *must be decoded to another codeword. However, the only word* x *such that* $D(032, x) = 1$ *and* $D(001, x) > 2$ *is* 132, *so* 132 *must be a codeword. Then* 112 *is decoded to* 132 *and all other words distance* 1 *from* 112 *cannot be codewords. Since this includes all words on the tetrahedron containing* 111, *by Theorem* 1 *we would not have a P1ECC. Therefore,* 001 *cannot be a codeword. The following table illustrates the procedure for showing the other non-codewords. If a word in the first column is assumed to be a codeword, then it decodes the word in the second column. This implies the word in the third column cannot be a codeword, and it must be decoded to the word in the fourth column. Since the word in the fourth column is then a codeword, it decodes the word in the fifth column, which implies the words in the sixth column cannot be codewords. Since the words in each row of the fifth and sixth columns make up the tetrahedrons in Theorem* 1, *we have a contradiction.*

codeword	decodes	non-codeword	decoded by	decodes	non-codeword
001	031	032	132	112	Δ110
002	032	031	231	221	Δ220
003	023	021	321	331	Δ330
110	120	123	023	003	Δ000
112	102	103	203	223	Δ220
113	103	102	302	332	Δ330
220	210	213	013	003	Δ000
221	201	203	103	113	Δ110
223	203	201	301	331	Δ330
330	320	321	021	001	Δ000
331	301	302	102	112	Δ110
332	302	301	201	221	Δ220

It follows from Theorems 1 and 2 that if there is a P1ECC on TH(4,3), then 000, 111, 222, and 333 must be codewords. However, the following table shows that this cannot be the case.

codeword C	000	111	222	333
	001	110	220	330
D1(C)	002	112	221	331
	003	113	223	332
	021	120	210	310
	031	130	230	320
non-codewords	012	102	201	301
D2(C)	032	132	231	321
	013	103	203	302
	023	123	213	312
	010	100	200	300
possible	011	101	201	303
codewords	020	121	211	311
D3(C)	022	122	212	313
	030	131	233	322
	033	133	232	323

No matter what combination of possible codewords we choose to cover D2(C), our new codewords will either be within distance 2 of each other, or non-codewords in D2(C) will not be covered. Thus, there is no P1ECC on the graph for TH(4,3).

2.42 P2ECC

If $a_3 = a_2 = a_1$ and $b_3 = b_2 = b_1$ but $a_3 \neq b_3$, then $D(a_3a_2a_1, b_3b_2b_1) = 5$. Since we may recognize this as distance = 2e + 1 in a code where e is the number of errors corrected, we may expect the code for TH(4,3) to correct 2 errors. Then, the most intuitive choice for codewords is 000, 111, 222, and 333 since they are farthest apart. However, the following table shows that these words will not generate a P2ECC.

codeword C	000	111	222	333
	001	110	220	330
D1(C)	002	112	221	331
	003	113	223	332
	021	120	210	310
	031	130	230	320
D2(C)	012	102	201	301
	032	132	231	321
	013	103	203	302
	023	123	213	312
	010	100	200	300
	011	101	201	303
non-codewords	020	121	211	311
D3(C)	022	122	212	313
	030	131	233	322
	033	133	232	323

Since the non-codewords are not covered by any codeword, we do not have a P2ECC. Note that we cannot choose any of the non-codewords as codewords because if we did, the words in D1 (C) and D2(C) will be distance 2 from 2 different codewords.

Theorem *TH*(4,3) *does not have a P2ECC.*

Proof *For a P2ECC, every word is either a codeword or has distance* ≤ 2 *from exactly one codeword. Thus, for* 000, 111, 222, *and* 333, *we need to choose codewords that are distance* ≤ 2 *from each word. That is, in Table* 1, *we must choose one word from each column as a codeword. Notice that the words in each column are of one of the following forms: aaa, aab, and abc. Define* $A = \{0,1,2,3\}$, *and let* $a \in A$. *Define* $B = A \backslash a$, *and let* $b \in B$. *Define* $C = B \backslash b$, *and let* $c \in C$. *Also,* *define D = C\c, and let d ∈ D.*

- **Case (aaa is a codeword)** *Now, suppose we choose a word aaa to be a codeword (that is,* 000, 111, 222, *or* 333). *Table* 2 *shows the outcomes of this choice. Table* 3 *shows distance between each of the possible codewords from table* 2. *Since each of the possible codewords are less than distance* 5 *apart, we may only choose one as a codeword. However, there is no single possible codeword that is distance* ≤ 2 *from the others, so no choice will cover all words. Thus, we will not have a P2ECC if aaa is a codeword.*

	Table	1	
000	110	220	330
001	111	221	331
002	112	222	332
003	113	223	333
012	102	201	301
013	103	203	302
021	120	210	310
023	123	213	312
031	130	230	320
032	132	231	321

Table 2				
Codeword:	aaa			
covers:	aab	abc	acb	adb
	aac	abd	acd	adc
	aad			
non-codewords	Δbc0	aba	bbc	bac
	Δbd0	abb	bbd	bad
	Δcb0	aca	ccb	cab
	Δcd0	acc	ccd	cad
	Δdb0	ada	ddb	dab
	Δdc0	add	ddc	dac
possible codewords	bba	bbb	baa	bab
	cca	ccc	caa	cac
	dda	ddd	daa	dad

	Table 3											
	bba	bbb	baa	bab	cca	ccc	caa	cac	dda	ddd	daa	dad
bba	0	1	3	3	3	4	4	4	3	4	4	4
bbb	1	0	3	3	4	5	4	4	4	5	4	4
baa	3	3	0	1	4	4	1	2	4	4	1	2
bab	3	3	1	0	4	4	2	3	4	4	2	3
cca	3	4	4	4	0	1	3	3	3	4	4	4
ccc	4	5	4	4	1	0	3	3	4	5	4	4
caa	4	4	1	2	3	3	0	1	4	4	1	2
cac	4	4	2	3	3	3	1	0	4	4	2	3
dda	3	4	4	4	3	4	4	4	0	1	3	3
ddd	4	5	4	4	4	5	4	4	1	0	3	3
daa	4	4	1	2	4	4	1	2	3	3	0	1
dad	4	4	2	3	4	4	2	3	3	3	1	0

Case (aab is a codeword) *Similarily, if we choose a word aab as a codeword, Table* 4 *shows the outcome and Table* 5 *shows the words covered by each possible codeword. Again, each possible codeword is less than distance* 5 *from the others, so we may only choose one. However, none of them cover all of the non-codewords in Table* 4*, so we will not have a P2ECC if a word aab is a codeword.*

Table 4					
codeword:	aab				
covers:	Δaa0	abc			
	Δac0	abd			
	Δad0	cdb			
		dcb			
non-codewords:	Δbc0	Δdb0	aba	bac	cdd
	Δbd0	Δdd0	abb	bad	dca
	Δcb0	Δca0	bbc	cda	dcd
	Δcc0	Δda0	bbd	cdc	dcc
Possible Codewords:	bba	bbb	baa	bab	

Table 5			
bba	bbb	baa	bab
Δbb0	Δbb0	Δba0	Δba0
Δbc0	bac	Δca0	Δbc0
Δbd0	bad	Δda0	Δbd0
bac	bcd	bbc	bbc
bad	bca	bbd	bbd
cda	bdc	bcd	caaa
dca	bda	bcb	daa
		bdc	cad
		bdb	dac

Case (abc is a codeword) *Now, we may choose a word abc as a codeword. Again, Table* 6 *shows the outcome. Since there are no words distance* 5 *from a word of the form abc, abc cannot be a codeword.*

Table 6			codeword:	abc							
covers	Δab0	Δaa0	Δad0	Δdb0	aca	acd	ddc	dac	bdc	cbd	
D3	acb	cba	dda	ddb	ddd	bda	bdb	bdb	bbc	bac	bcd
	acc	cbc	dab	daa	dad	ddc	cad	cdb	dca	dcd	
D4	dcb	bba	bbb	bbd	baa	bab	bad	bca	bcb	bcc	
	dcc	cda	cdc	cdd	caa	cab	cac	cca	ccb	ccc	

Thus, TH(4,3) does not have a P2ECC. However, it does have a trivial error-detecting code if we choose 000, 111, 222, 333 as codewords. Then, any word not of the form aaa will be detected as a non-codeword. In fact, for TH(k,n), if we choose $a_n a_{n-1} ... a_0$ of the form aaa where $a \in \{0, 1, ..., k\}$ as a codeword, then we will always have this trivial error-detecting code.

Conclusion

Even though the generalized Towers of Hanoi with four pegs does not seem to produce good one- or two- error correcting codes, its symmetry and patterns could attract more study. It may be enjoyable to investigate other values of k and n and try to prove general information about the graph for TH(k,n) as well as perhaps find a gerneral recursive construction for the graphs. Also, one might explore distance further by constructing a distance formula for any two words. This is an open-ended problem for many codes, and studying it would surely be a worthwhile activity.

References

(1)P. Cull and I. Nelson. *Error-Correcting Codes on the Towers of Hanoi Graphs*. Technical Report-NSF Grant DMN 93-00-281. Department of Computer Science, Oregon State University. Corvallis, Oregon.

(2)P. Cull and David Bode. *Alternate Labelings for Graphs Representing Perfect-One-Error-Correcting Codes*. Technical Report-NSF Grant. Department of Computer Science, Oregon State University. Corvallis, Oregon.

(3)A.M. Hinz. *An Iterative Algorithm for the Tower of Hanoi with Four Pegs*. Computing 42, 133-140 (1989).