Coding theory in a life or death problem

IB Extended Eassay

Pau Cantos Coll

Course 2018-2019-2020

Contents

0 Introduction

0.1 Problem and Approach

Before starting the IB course, I received an enigma which we transcribe as follows:

Problem 1. *There are two prisoners in a cell and a guard. The guard is about to execute them but offers a last chance to survive:*

—In the room next door there is a chess table with a coin facing heads or tails in each of the squares —says the guard—. You, of course, know nothing about the configuration of the coins, which can be any. What I will do is to take one of you two to the room and tell him the ***square of life*** *(any of the board). He then will have to turn exactly one of the coins of the table (meaning changing it from heads to tails or from tails to heads). Then I will take the other prisoner to the room (without communication with the first) and will have to guess which is the square of life.*

—If he guesses correct you will both be released and else be executed. Now you have some time to argue the strategy.

What is the best strategy they can play (meaning the one that gives them more chance to survive)?

Similarly as Pólya did in his famous book [8]. We present this method to try to solve this kind of enigma-strategy-combinatorics problems:

1. **Reformulation**. Understand the problem, then express it using existing mathematical concepts. We will have to define concepts and turn to review or discover new theory. This step is done mainly by applying logic: distinguish what is relevant and which implications has.

2. **Generalization**. See whether there is some way to enlarge the scope of the problem. Would the particular solution follow from the solution of this generalization? Can one make it the reciprocal also true? If there are *smaller* cases of the statement, we solve those first to find ideas, patterns and claims. Considering always the general result might be useful for example if the easiest solution uses induction.

3. **Development**. Once we have some patter to proof or just a reformulation from which something interesting might be derived, develop these ideas. In this step, we will have to apply any known mathematical method or theorem. Always look for coincidences to have brainwave ideas. If we have a claim or an intuition that is not proven, we just assume it and see if the answer can follow from there.

4. **Similar smaller problem**. Find a similar easier problem and attack it using this same method.

5. **Research and Applications**. When we are done or else we threw in the towel, look to the solution or if it isn't published use your reformulation to find any information on the literature of mathematics that can help solve the problem. Finally, reflect on your work and see where it can be applied to real-life problems or to solve other problems of this kind and which new questions posted your solution.

The order of the steps is not strict, one can jump alternately between steps 2 and 3 depending on the patience.

We will soon discover that these steps conduce us to coding theory: a branch of mathematics and combinatorics. Therefore the title of the work and the following research question:

Question 1. *To what extent can the accurate analysis of all possible ways to solve this problem be used to learn coding theory and set out new questions and perspectives in this branch of mathematics?*

The later steps approximately correspond to the main sections of the work so now that it is clear what is done on each, we will just get to the point.

0.2 Notation

Since we will have to use some notation, not all standard or necessarily known. Consult this table for a compilation of accepted notation.

For any structures (i.e. set, group, graph, etc.) A and B supporting an operation ($a * b$ between any $a \in A, b \in B$), natural numbers k and p prime, function f with A in the domain and vector space V of dimension $n > k$, we have:

Expression	Description
$A * B$	Structure $\{a * b \vert a \in A, b \in B\}$ (preserving properties).[1]
$A \times B$	Cartesian or direct product: $\{(a, b) \vert a \in A, b \in B\}$ (preserving properties).[2]
A^k	$A^k = \underbrace{A \times A \times \cdots \times A}_{k\ A\text{'s}}$ with the $\times$ product defined above.
$\vert A \vert$	Cardinality or order, i.e. number of elements of the structure.
$f(A)$	The image of A under f: $\{f(a) \vert a \in A\}$ (preserving properties) When f is of type $f(a) = a + b$, $f(A)$ is called coset of A.
2^A	Set of subsets: $\{S \vert S \subseteq A\}$ (preserving properties).
$\binom{A}{k}$	A choose k, $\{S \in 2^A \vert \vert S \vert = k\}$ (preserving properties).
$[k]$	$\{0, 1, 2, \cdots, k-2, k-1\} = \{n \in \mathbb{N} \vert n < k\} = [0, k) \cap \mathbb{N}$
e_k, $\tilde{e}_k$	$e_k := (\underbrace{0, 0, \cdots, 0}_{k\ 0\text{'s}}, 1, 0, \cdots, 0)$ and $\tilde{e}_k$ the same but in column form, in space V
$\mathbb{Z}_k$	The group formed by integers with addition modulo k: $\mathbb{Z}/k\mathbb{Z} = \mathbb{Z}/(k) = \mathbb{Z}/\dot{k}$.[3]
$\mathbb{F}_p$	The field formed by integers modulo p with addition and product.[4]
S_k	The group of permutations of $[k]$, with the operation of composition ($\circ$).

Table 1: Accepted notation.

Understand that if A and B were numbers, the first 7 notations apply as it is usual.

[1] This notation is not introduced by the author but inspired on the problem A2 of the 2012 IMO Shortlist.
[2] If one is not familiar with it see theory section A.4.
[3] If one is not familiar with this group see the theory section A.1.
[4] If one is not familiar with this field see the theory section A.1.

1 Reformulation

Observation 1.1. *We can assume there is no random component in the strategy of the prisoners, else, they would rather predetermine their decision getting no worse results (we always assume both prisoners are infinitely smart and the do never get confused). Thus, since the only information that the second prisoner has on entering the room is just what he sees in the table, its choice uniquely depends on this. What we are describing is a function. This second player receives a configuration of the table as input and returns a uniquely determined square (guess of square of life) as output.*

We need to choose how to mathematically represent the possible states of the board and other things. So first, let us assign numbers to things so that we can speak more mathematically.

- Let n denote the number of squares on the board (in our particular case $n = 8 \times 8 = 64 = 2^6$).[5]
- Assign numbers 0 and 1 to head and tail respectively.[6]
- Assign a number from the set $[n]$ to each of the n squares of the board as seen in Figure 1 (which can be distinguished by the letters and numbers on the borders).[7]

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Figure 1: Canonical assignation of the numbers 0 to 63, to the squares of the board.

The last two assignations will be called *canonical*.

In the table we have n entries each with a value form the set $\{\text{head}, \text{tail}\} \leftrightarrow \{0, 1\}$. Thus, a certain state of the table can be seen as a binary vector, word or tuple: the set of possible states of the board is represented by: $\{0, 1\}^n$. Let $m := |\{0, 1\}^n| = |\{0, 1\}|^n = 2^n$ be the number of possible configurations of the board. Finally we can say that $F_n : \{0, 1\}^n \to [n]$ is *the function that the second prisoner plays*, in the sense we explained in observation 1.1. Once the prisoners have defined their function F_n, the first one knows exactly what the second is going to say once he chooses a coin to turn. Hence he will just try to change the *image* of the board under F_n to the correct square of life. There are only two possibilities: either there is no way to turn a coin such that the F_n maps to the correct square of life, in which case they are doomed to die, or there is some way to do it and then they are immediately free men. Thus there is no secret in how the prisoners must play once they agree in F_n. That's why we will directly call the function F_n: the **strategy**.

A strategy that allows them to survive *always* will be called **safe**. From now, our aim is to determine weather there is any safe strategy, and, if there is, and we can find it, we are done.

Now let us define how a *turn* of coin is described in our mathematical formalization. Here is where we can decide between some branches of math so we distinguish three different formulations:

[5] See generalization on section 2.1.
[6] See generalization on section 2.2.
[7] See variation on section 3.1.

1.1 Algebraic formulation

The set $\{0,1\}$ is viewed as the elements of $\mathbb{F}_2$. Like this, a coin in state $c \in \{0,1\}$ is in state $c+1$ *when turned upside down.*

For operating on the whole table:

Definition 1.1. *Let $G_n := \mathbb{F}_2^n$ be the **vector space** representing the possible configurations of the board,*[8] *with $B_n := \{e_0, e_1, \cdots, e_{n-1}\}$ being its canonical* basis.

Like this, if the state of the board is represented by a configuration $v \in G_n$, then after turning coin in square i, the state turns to $v + e_i$. Then formally a safe strategy is an $F_n : G_n \to [n]$ such that:

$$\forall v \in G_n, \forall g \in [n], \exists i \in [n], F_n(v + e_i) = g \tag{1}$$

Formulation 1.1. *There are exactly n possible turns from any configuration $v \in G_n$ (namely $c + e_i$ for $i \in [n]$) and we have to be able to reach each of the n possible squares of life (if we want to be always save). Thus there is exactly one turn for each square of life. In other words, the function $\sigma_v : [n] \to [n]$ defined as $\sigma_v(i) := F_n(v + e_i)$ needs to be surjective. For any application $\sigma : [n] \to [n]$ the following are equivalent expressions:*

1. *σ is injective.*
2. *σ is surjective.*
3. *σ is bijective.*
4. *σ is a permutation of $[n]$.*
5. *$\sigma \in S_n$.*

Thus all this apply to σ_v for all $v \in G_n$.

Summarizing, (1) is equivalent to:

$$\forall v \in G_n, \sigma_v \in S_n \tag{2}$$

1.2 Graph theory formulation

Definition 1.2. *Let G_n be the **graph** that has the states of the board $\{0,1\}^n$ as vertices and an edge from u to v iff one can turn a coin from state u to get to state v.*

Observation 1.2. *G_n is the n-hypercube graph.*[9]

A function ϕ from the vertices of a graph to some finite set can be thought as a *colouration* of the graph, where each value of the codomain is associated with a colour.

Definition 1.3. *Let $\mathcal{C}_i := \{v \in \{0,1\}^n | F_n(v) = i\}$, be the sets of vertices "of the same colour $i \in [n]$".*

Formulation 1.2. *In this new scope formulation 1.1 is equivalent to saying that any vertex $v \in G_n$ has exactly one neighbour of each colour. This is also equivalent to the property that there must be no two vertices of the same colour at distance 2 in the graph.*

Taking it further, if we consider the distance-2 graph of G_n: namely G_n^2, then we are searching for a normal colouration of G_n^2 (normal meaning no two adjacent vertices with the same colour).

Observation 1.3. *There are $|\mathcal{C}_i| = \frac{m}{n}$ vertices of each colour.*

[8] If one is not familiar with vector spaces see theory section A.1

[9] If one is not familiar with the hypercube graph see theory section A.2.

Proof. To give a formal proof of this, that is very intuitive because each code has the same number of words, namely $\frac{1}{n}$ of the total, we can use a double counting of the set of pairs $\{(u,v) \in \mathcal{C}_i \times G_n | v, u \text{ adjacent}\}$, using observation 1.2:

$$\sum_{v \in G_n} \sum_{u \in \mathcal{C}_i \text{ adjacent } v} 1 = |\{(u,v) \in \mathcal{C}_i \times G_n | v, u \text{ adjacent}\}| = \sum_{u \in \mathcal{C}_i} \sum_{v \text{ adjacent to } u} 1$$

$$\sum_{v \in G_n} 1 = \sum_{u \in \mathcal{C}_i} n$$

$$m = |G_n| = n|\mathcal{C}_i|$$

$$|\mathcal{C}_i| = \frac{m}{n}$$

□

1.3 Coding theory formulation

Coding theory has developed some notation very useful here, one can find explanation in theory section A.3.1, so read it first.

Definition 1.4. *Let G_n be the n-dimensional Hamming space over $\mathbb{F}_2$.*

As we saw in previous section in definition 1.3, the function F_n splits the set $\{0,1\}^n$ in n codes, namely $\mathcal{C}_i$. The Hamming distance of two words is the same as the distance between the corresponding vertices in the graph, in other words, the graph G_n is "equivalent" to the space G_n (that's why we call them all the same).

Revising what we have from the last reformulation:

Formulation 1.3. *If two of the following three properties on a code $\mathcal{C}$ are fulfilled, then the code is called **safe** and the third one is also fulfilled by free.*

1. ***Size**: The code has $|\mathcal{C}| = \frac{m}{n}$ elements.*
2. ***Forbidden distance 2**: No two words of $u, v \in \mathcal{C}$ have distance $d(u,v) = 2$ or equivalently there is no word $w \in G_n$ having two distinct neighbours $w \in \mathcal{C}$.*
3. ***Dominant**: Any word is at distance 1 of a word in $\mathcal{C}$.*

*Then finding a safe strategy reduces to finding a partition of the Hamming space into safe codes, which we will call a **safe partition/colouration**.*

Proof. If a code $\mathcal{C}$ satisfies properties...

1. 2 and 3 then each vertex $v \in G_n$ has exactly one neighbour $u \in \mathcal{C}$ and property 1 is also satisfied by the double counting argument used in proof of 1.3.
2. 1 and 3 then again counting each vertex must be dominated no more than once from which property 2 follows.
3. 1 and 2 then any vertex is dominated at most once and again counting all are dominated, i.e. property 3 is fulfilled.

□

After all this formulations, a good question is: "Which of the preceding three formulations will we use? Definition 1.1, 1.2 or 1.4 for G_n?". And the answer is: "Not a particular one but all at the same time". This means we will use arithmetic properties of $\mathbb{F}_2$ to model the movements, talk about the function as a colouration of the graph, work with codes, distance and the Hamming weight, use vector-matrix properties, etc.

For example let us for a moment combine the algebraic and coding theory formulations to give a slightly different perspective:

Definition 1.5. *Let the* **k-*stage*** *of G_n be $I_k^n := \{w \in G_n | \text{wt}(w) = k\}$, i.e. the set of binary words with k out of n ones.*

Observation 1.4. *The property of forbidden distance 2 is equivalent to $\mathcal{C} - \mathcal{C} = \mathcal{C} + \mathcal{C}$ disjoint to I_2^n.*

2 Generalization

There are a couple of obvious generalizations of the problem that one may have already noticed.

2.1 Size of board

Can we ask the same problem for a general n (other than 64)? Is the problem really similar for all values of $n \in \mathbb{N}$? The answer is: "Yes, the problem can be generalized, but no, it is not the same for all n". Consider the following argument:

Observation 2.1. *For there to be a safe strategy on a board of n squares, it is necessary that $|\mathcal{C}_i| = \frac{m}{n} \in \mathbb{N}$ so we must have $n|m = 2^n$.*[10] *If n divides a power of 2 it must be a power of 2, namely $n = 2^N$. Then clearly $N < 2^N = n$ for all $N \in \mathbb{N}$ so $|\mathcal{C}_i| = \frac{m}{n} = \frac{2^n}{2^N} = 2^{n-N} \in \mathbb{N}$ is satisfied (in our first case $n = 64$ $N = 6$ and $|\mathcal{C}_i| = 2^{64-6} = 2^{58}$, very large).*

It is not that the problem can't be asked for n not a power of 2 but there won't be safe strategies there so we just focus in powers of 2.

2.1.1 Small case analysis

Now that we know which smaller cases to consider, let's do what we declared in our first planning, solve the smaller cases basically by trying all the possibilities.

Case 0. *For the trivial case $N = 0$, $n = 1$, $m = 2$, there should be only 1 code of length 2 from the set $G_1 = \{0, 1\}$.*

There is 1 solution with F_1:

v	$F_1(v)$
0	0
1	0

Table 2: Solution for F_1 safe

Using the code:

$$\{0, 1\}$$

Case 1. *For the case $N = 1$, $n = 2$, $m = 4$, there must be 2 codes of length 2 from set $G_2 = \{00, 01, 10, 11\}$.*

There are 4 solution of F_2:

v	$F_2(v)$
00	0
01	0
10	1
11	1

v	$F_2(v)$
00	1
01	1
10	0
11	0

v	$F_2(v)$
00	0
01	1
10	0
11	1

v	$F_2(v)$
00	1
01	0
10	1
11	0

Table 3: Solutions for F_2 safe

Using the codes:

$$\{00, 01\} \quad \{00, 10\}$$
$$\{10, 11\} \quad \{01, 11\}$$

Case 2. *For the case $N = 2$, $n = 4$, $m = 16$, there must be 4 codes of length 4 from set $G_4 = \{0000, 0001,$*
$0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$.

There are 96 solution of F_4, that we will not write but one would have to be able to construct once has read section 3.1 and use the code:

$$\{0000, 0001, 1110, 1111\}$$

[10] $a|b$ means a divides b.

Case 3. *For the case $N = 3$, $n = 8$, $m = 256$, there must be 8 codes of length 32.*

A computer search optimized with the use of the theory on section 3.1 shows that there are 7200 elementary codes or 28800 codes in total which can form at least 290304000 safe strategies. The explanation of the algorithm and the output can be seen in annex B.

For if it has not become clear already, we are going to make the conjecture that cannot wait for more:

Claim 2.1. *For $n = 2^N$ ($N \in \mathbb{N}$), there exists a safe strategy F_n.*

The obvious way to prove it would be just analyzing the cases above to find a pattern that can be generalized. Unfortunately, it is unlikely that one sees the generalization from here (I didn't the first time). Thus the proof will come in the development when we are prepared.

2.2 Different coin

The second generalization consists of changing the coin for another random object such as a die, i.e. with another number q of faces. If the first prisoner is asked to increase the number of a face by 1, then the same strategy that with coins can be applied, considering like 0 or 1 an even or odd number on the face of a die. Else if the first prisoner is allowed to chose a square and change the visible face that square to any of the other ones, since the number of choices of the first prisoner increases, the difficulty of the game might be increased by the guard, for example, the second prisoner will have to tell the square of life and the movement of the first prisoner meaning $a - b$ where a and b are the initial and final positions of the die moved by the first prisoner (with operation modulo q), or just something from a set of $n(q - 1)$ elements.

We see that these generalizations tend to be less natural and, believe me, they do not increase much the difficulty of the problem once we know the solution for our case $q = 2$ and apply the same with operations in $\mathbb{F}_q$. This type of generalization will be more interesting in another similar problem.[11]

[11] See section 4

3 Development

3.1 Trivial transformations

By analysis of the different solutions on the cases seen in the last section, or otherwise by simple reasoning, one may realize that there is repeated information because there are trivial variations of F_n and of the codes that conserve properties.

Observation 3.1 (**Color permutation**). *Imagine one found a safe partition of G_n. Then one can colour each of them with whichever colour and in any case find a safe colouration F_n. In other words, if F_n is a safe strategy then $F_n \circ \sigma$ is also a safe strategy for any $\sigma \in S_n$.*

This makes us realize that for an efficient search it is easier to first find ordered safe partitions of G_n and then consider the $|S_n| = n!$ trivial variations.

3.1.1 Automorphisms of the Hamming space

Definition 3.1 (**Translation**). *Let $t_v : G_n \to G_n$ be defined as doing a translation (modulo 2) of a vector $v \in G_n$, i.e.:*

$$t_v(c) := c + v \tag{3}$$

Definition 3.2 (**Entry permutation**). *Let $\psi_\sigma : G_n \to G_n$ be defined as doing a permutation $\sigma \in S_n$ on the entries of each vector $v \in G_n$. Note that this variation can be seen as multiplying by a matrix:*

$$\psi_\sigma(v) := vT_\sigma \tag{4}$$

where T_σ is a square $n \times n$ matrix defined as $T_\sigma := \sum_{i=0}^{n-1} \tilde{e}_{\sigma(i)} e_i$.

Let's explore the properties of t and ψ.

$$t_a \circ t_b = t_{a+b} \tag{5}$$
$$\psi_\sigma \circ \psi_\mu = \psi_{\sigma\circ\mu} \tag{6}$$

This means t and ψ are invertible:

$$t_v^{-1} = t_{-v} \tag{7}$$
$$\psi_\sigma^{-1} = \psi_{\sigma^{-1}} \tag{8}$$

which also means they are bijective. Furthermore by (3) and (4):

$$t_v(a) + t_v(b) = (a + v) + (b + v) = a + b \tag{9}$$
$$\psi_\sigma(a) + \psi_\sigma(b) = aT_\sigma + bT_\sigma = (a + b)T_\sigma = \psi_\sigma(a + b) \tag{10}$$

and since:

$$\mathrm{wt}(\psi_\sigma(x)) = \mathrm{wt}(x) \tag{11}$$

then also:

$$d(t_v(a), t_v(v)) = \mathrm{wt}(t_v(a) + t_v(b)) = \mathrm{wt}(a + b) = d(a, b) \tag{12}$$
$$d(\psi_\sigma(a), \psi_\sigma(v)) = \mathrm{wt}(\psi_\sigma(a) + \psi_\sigma(b)) = \mathrm{wt}(\psi_\sigma(a + b)) = \mathrm{wt}(a + b) = d(a, b) \tag{13}$$

Summarizing:

Observation 3.2 (**Automorphisms**). *Both t and ψ are automorphisms of the Hamming space, meaning that they are one-to-one functions that preserving distance. This also implies that $\mathcal{C}$ is a safe code iff $(t_v \circ \psi_\sigma)(\mathcal{C})$ is also safe for any $v \in G_n$ and $\sigma \in S_n$.*

Observation 3.3 (**Partition using cosets**). *For a safe code $\mathcal{C}$, the family of cosets $\mathcal{C}_i = t_{e_i}(\mathcal{C})$ for $i \in [n]$ forms a safe partition of G_n.*

Proof. We already know by observation 3.2 that all $\mathcal{C}_i$ are also safe. We now need to prove that they are pairwise disjoint. Suppose by contradiction that $\mathcal{C}_i \cap \mathcal{C}_j \neq \emptyset$ because:

$$\mathcal{C}_i \ni a + e_i = b + e_j \in \mathcal{C}_j$$

for $a, b \in \mathcal{C}$ and $i \neq j \in [n]$. Then also:

$$a - b = e_j - e_i$$

so

$$d(a,b) = \mathrm{wt}(a-b) = \mathrm{wt}(e_j - e_i) = d(e_j, e_i) = 2$$

Which is a contradiction if $\mathcal{C}$ is safe. Finally, by injectivity of t, $|\mathcal{C}_i| = |\mathcal{C}| = \frac{m}{n}$ and the union $\bigcup_{i=0}^{n-1} \mathcal{C}_i$ has $n\frac{m}{n} = m$ elements, all G_n. □

With this we see that *it is sufficient to find* ***one*** *safe code to find a strategy function*!

3.1.2 Classification and counting of variations

Definition 3.3. *If $e_i \in \mathcal{C}$ we say that $\mathcal{C}$ is* ***of type*** **i**.

Observation 3.4. *I_1^n is the set of neighbours of 0 in G_n so by formulation 1.1 there is exactly one of those belonging to $\mathcal{C}$. This means the* type *of a code is unique.*

Now if we want to do an efficient search, we shall explain the following reasoning:

Definition 3.4. *We call a safe code $\mathcal{C}$* ***elementary*** *iff $0 \in \mathcal{C}$.*

Definition 3.5. *We call an elementary safe code of type 0:* ***primitive.***

Consider any safe code $\mathcal{C}$ of type i and consider the coset $\mathcal{C}_0 := t_{e_i}(\mathcal{C})$ to be its **boss**, which is *elementary* because contains $t_{e_i}(e_i) = 0$. Conversely from any elementary safe code $\mathcal{C}_0$ of type t, consider the set of cosets $\mathcal{C}_0 + e_i$ (constructed as in 3.3) to be its **employee**.

Similarly let $c \in S_n$ be the cyclic permutation defined as $c(i) = i + 1$ (modulo n, i.e. $n = 0$). Then for any elementary safe code $\mathcal{C}$ of type i consider $\psi_c^{-i}(\mathcal{C})$ to be its **parent**, which is *primitive* because contains $e_0 = \psi_c^{-i}(e_i)$. Conversely from any primitive safe code $\mathcal{C}_0$, consider the set of cosets $\psi_c^i(\mathcal{C}_0)$ to be its **child**.

Definition 3.6. *Let $\mathcal{C}_0 R_1 \mathcal{C}$ express the relation $\mathcal{C}_0$* ***is boss of*** *$\mathcal{C}$ or $\mathcal{C}$* ***is employee of*** *$\mathcal{C}_0$. Similarly R_2 for the* ***relation parent-child****.*

Observation 3.5. *Any safe elementary code has exactly n employee. Any code has exactly one boss. Similarly for the relation parent-child.*

Now we have:

$$2^{G_n} \supseteq \binom{G_n}{m/n} \supseteq C_n \supseteq E_n \supseteq P_n \tag{14}$$

where C_n is defined as the set safe codes in G_n, E_n the elementary ones, and P_n the primitive ones.

Observation 3.6. *By a double counting on the set of pairs $\{(\mathcal{C}_0, \mathcal{C}) \in E_n \times C_n | \mathcal{C}_0 R_1 \mathcal{C}\}$ and using observation 3.5:*

$$\sum_{\mathcal{C}_0 \in E_n} \sum_{\mathcal{C} \text{ employee of } \mathcal{C}_0} 1 = |\{(\mathcal{C}_0, \mathcal{C}) \in E_n \times C_n | \mathcal{C}_0 R_1 \mathcal{C}\}| = \sum_{\mathcal{C} \in C_n} \sum_{\mathcal{C}_0 \text{ boss of } \mathcal{C}} 1$$

$$\sum_{\mathcal{C}_0 \in E_n} n = \sum_{\mathcal{C} \in C_n} 1$$

$$n|E_n| = |C_n|$$

This argument is correct even thought an elementary code could be boss and employee of itself. The analogous argument for the relation parent-child shows that $|E_n| = n|P_n|$ and thus $|C_n| = n^2|P_n|$.

To count and organize the set of solutions clearly we say that:

Definition 3.7. *A partition of G_n in safe codes is called* ***normal*** *if it can be constructed as the family of cosets of a code $\mathcal{C}$ as described in observation 3.3.*

The question that we need to ask is weather all the possible partitions of G_n in safe codes are normal. This would completely reduce the search of strategies to the search of safe codes or even more to elementary codes. Thus we conjecture it:

Conjecture 3.1. *Any partition of G_n into safe codes is* normal.

If the conjecture above was true we would have that the number of solutions for F_n is $n!\,|C_n| = n!\,n^2|P_n|$. Further ideas in this direction, such as only searching for *equivalence classes* closed by linear transformations t and ψ could lead to sharper results and faster algorithms to find all possible solutions, this ideas are explored in [5].

3.2 Linear codes

We hope all this ideas conduce the reader to naturally consider the following idea:

Definition 3.8. *A code $\mathcal{C}$ is called **linear** iff it is closed under addition, meaning that for all $a, b \in \mathcal{C}$, $a + b \in \mathcal{C}$.*[12]

Let's analyze some properties of a liner code $\mathcal{C}$ in relation to what we have:

Observation 3.7. *If a code is linear and nonempty then it contains $a + a = 0 \in \mathcal{C}$ (for any $a \in \mathcal{C}$). Also since the inverse of any element in G_n is itself, $(\mathcal{C}, +)$ is a normal subgroup of G_n.*

Lemma 3.1. *Any Abelian group $(H, +)$ with all the elements of order 2, i.e. $h+h = 0 := \mathrm{id}_+$ for every $h \in H$ is isomorphic to $\mathbb{Z}_2^k$, being $|H| = 2^k$.*

Proof. By the fundamental theorem of Abelian groups A.1, we can say $H \cong \bigtimes_{i=1}^{k} \mathbb{Z}_{a_i}$ (where $\bigtimes$ represents the operation $\times$ over a sequence of elements). Then for each $i \in [k]$ take the element $h_i \in H$ corresponding to $e_i \in \bigtimes_{i=1}^{k} \mathbb{Z}_{a_i}$ (by the isomorphism) to get that $e_i + e_i$ corresponds to $h_i + h_i = 0$ so $a_i = 2$. Therefore 3.1 follows taking into account that $|H| = |\mathbb{Z}_2^k| = |\mathbb{Z}_2|^k = 2^k$. □

Observation 3.8. *If $\mathcal{C}$ is linear then $\mathcal{C} + \mathcal{C} = \mathcal{C}$ by definition and thus by observation 1.4, $\mathcal{C}$ is safe iff it is disjoint to I_2^n and has m/n elements.*

Observation 3.9. *The family of cosets $\mathcal{C}_0, \mathcal{C}_1, \cdots, \mathcal{C}_{n-1}$ defined in 3.3 from a linear code $\mathcal{C}$ is the family of* congruence classes of the quotient *$G_n/\mathcal{C}$ because they are* the *$|G_n/\mathcal{C}| = |G_n|/|\mathcal{C}| = n$* different *cosets of $\mathcal{C}$ (by Lagrange's). Hence F_n is analogous to the canonical mapping $G_n \to G_n/\mathcal{C}$, i.e. can be considered an homomorphism from G_n to $([n], *)$ as a group isomorphic to the quotient $G_n/\mathcal{C}$ (being $*$ the induced operation). By lemma 3.1, $G_n/\mathcal{C}$, and thus also $([n], *)$, are isomorphic to $\mathbb{Z}_2^N$.*

With this we can finally describe the strategy function F_n that proves claim 2.1.

Definition 3.9. *Let $\rho_0 : \{0,1\}^N \to [n]$ be the canonical function mapping from a binary vector x to the number that represents in base 2:*

$$\rho_0(x) = xb_N$$

where the vector $b_N := \sum_{i=0}^{N-1} 2^i \tilde{e}_i$ (now not operating modulo 2).

In view of the variation 3.1, one can assume ρ_0 is the isomorphism between $\mathbb{Z}_2^N$ and $([n], *)$ then indeed $*$ is now exactly as XOR in the computers. Then we can redefine:

Definition 3.10. *Let $F'_n : G_n \to G_N$ as:*

$$F'_n = F_n \circ \rho_0^{-1}$$

i.e. mapping directly to the binary vector representation of the number assigned to the square.

See diagram below for support of what we are doing with the functions.
Like this we directly get the homomorphism condition as:

$$F'_n(x + y) = F'_n(x) + F'_n(y) \tag{15}$$

for any $x, y \in G_n$.

Also, by the automorphism ψ, we can assume $F_n(e_i) = i$ or $F'_n(e_i) = \rho_0^{-1}(i)$. Then finally, for any $x = (x_0, x_1, \cdots, x_{n-1}) = \sum_{i=0}^{n-1} x_i e_i \in G_n$, using 15:

$$F'_n(x) = F'_n\left(\sum_{i=0}^{n-1} x_i e_i\right) = \sum_{i=0}^{n-1} x_i F'_n(e_i) = \sum_{i=0}^{n-1} x_i \rho_0^{-1}(i) = xM_n \tag{16}$$

[12] This notion already existed, we didn't not invented it, see: [2, 3].

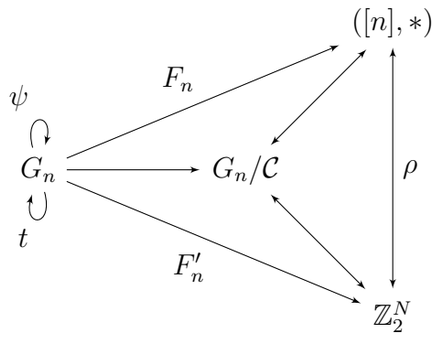

Figure 2: Graph of the homomorphism between our groups. The arrows represent homomorphism, or isomorphisms if they are directed, with their names.

where the $n \times N$ matrix $M_n := \sum_{i=0}^{n-1} \rho_0^{-1}(i)\tilde{e}_i$. For example:

$$M_4 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

If one wants to recover F_n, plugging definition 3.10 into 16:

$$F_n(x) = \rho_0(xM_n) \tag{17}$$

Take a moment to breath and explain this in normal words. What the prisoners basically do to compute F_n is just: take the numbers assigned to each of the squares of the board that have tail and do XOR of all of them. Then clearly what the second prisoner needs to do is evaluate $F_n(x)$ of the current configuration x of the board and then chose i such that $F_n(x + e_i) = g$ being g the square of life. Because $F_n(x + e_i) = F_n(x) * i$, his choice needs to be exactly $i = F_n(x) * g = F_n(x + e_g)$ and like this they always survive. We proved claim 2.1 and solved the general, and hence the particular problem 1 for the prisoners!!

How do the variations apply here? Let's undo the assumptions. The isomorphism ρ may be any function from G_N to the set $[n]$, thus $\rho = \rho_0 \circ \tau$ for any $\tau \in S_n$. Also the values $F_n(e_i)$ can be any permutation of $[n]$, thus $F_n(e_i) = \sigma(i)$, which is equivalent to:

$$i = F_n(e_{\sigma^{-1}(i)}) = F_n(\psi_{\sigma^{-1}}(e_i)) = F_n(\psi_\sigma^{-1}(e_i)) = \left(F_n \circ \psi_\sigma^{-1}\right)(e_i)$$

If we redefine the solution given in 16 as F_n^0 doing a similar reasoning as there we have $\psi_\sigma^{-1} \circ F_n \circ \rho^{-1} = F_n^0 \circ \rho_0^{-1}$ so we may get the complete set of normal solutions using linear codes as:

$$F_n = \psi_\sigma \circ F_n^0 \circ \rho_0^{-1} \circ \rho = \psi_\sigma \circ F_n^0 \circ \rho_0^{-1} \circ \rho_0 \circ \tau = \psi_\sigma \circ F_n^0 \circ \tau \tag{18}$$

for some permutations $\sigma, \tau \in S_n$. So, as one can see, all the variations are given by using colour and entry permutations. Also as applying ψ is like multiplying by a binary matrix one can indeed consider the variation ψ as permuting the columns of M_n.

One may also ask "Why don't we also use translation automorphism t?" The answer is that it cancels out with τ: if one applies a translation t to a normal partition G_n with a linear construction one just gets a permutation of the same codes since translating t inside the group of equivalence classes $G_n/\mathcal{C}$ just ends with the same codes.

Definition 3.11. *The strategies obtained like this will be called* ***linear****.*

And how to recover all possible linear codes from all possible linear solutions? By definition $\mathcal{C}_0 = \ker(F_n^0)$, thus all linear codes are obtained by coordinate permutations of the null-space of M_n.

This form of generation of linear codes is well known in coding theory. Indeed, any linear code $\mathcal{C}$ of size 2^k in G_n can be identified with a *generating* $k \times n$ matrix $G(\mathcal{C})$ such that $\mathcal{C} = G_k G(\mathcal{C})$, i.e. $\mathcal{C}$ is its *image* and a *check* $n \times k$ matrix $H(\mathcal{C})$ such that $\mathcal{C}H(\mathcal{C}) = \{0\}$, i.e. $\mathcal{C}$ is its null-space (in our case $H(\mathcal{C}_0) = M_n$ and $k = N$).

Given the beauty of these solutions one can, conjecture the following:

Conjecture 3.2. *All safe strategies are linear.*

However, the conjecture turns to be false because, as the computer search shows B, up to case $n = 8$ non-linear safe codes appear for the first time.

3.3 Totally linear algebraic approach

Consider the following theorem:[13]

Theorem 3.2. *Let G be the binary $m \times m$ adjacency matrix of a simple m-vertex graph. Then suppose one has x_i insects in vertex i for each $i \in [m]$. Then construct the vector $x^{(0)} = (x_1, x_2, \cdots, x_n) = \sum_{i=0}^{n-1} x_i e_i$ which represents the state of the insects. Now imagine in one second every insect at vertex i is divided into $d(i)$ copies ($d(i)$ is the degree of the vertex) each of which travels to one of the neighbour vertex. Then after a second the new vector representing the state of the insects is:*

$$x^{(1)} = xG \tag{19}$$

Then its also easy to proof by induction that after k seconds the state is:

$$x^{(k)} = xG^k \tag{20}$$

Observation 3.10. *Using this, suppose one puts an insect in each of binary words $w \in \mathcal{C}$ for a code C. Then if v is the m-length vector representing the state, $\mathcal{C}$ being safe is equivalent to:*

$$vG_n = 1_m \tag{21}$$

Where now G_n is the adjacency of the n-hypercube graph, $1_k := \underbrace{(1, 1, \cdots, 1)}_{k\ 1\text{'s}} = \sum_{i=0}^{k} e_k$ and the operations are no longer modulo 2.

What makes this approach difficult is that for $n > 1$, G_n is singular, because there are many solutions as seen before, but one sill might want to find the rank, then the null-space $\mathcal{N}_n$ of G_n and finally using a particular solution $\mathcal{C}_0$, that we already have, find the intersection of the set of solutions $\mathcal{C}_0 + \mathcal{N}_n$ with the vector space G_n.

3.4 Stages analysis

In this section we will just count the number of codewords in each stage of G_n:[14]

Definition 3.12. *For a given code $\mathcal{C}$ and a number $k \in [n+1]$ let a_k be the number of elements of $\mathcal{C}$ in the k'th stage, i.e.:*

$$a_k := |\mathcal{C} \cap I_k^n| = |\{x \in \mathcal{C} | \mathrm{wt}(x) = k\}|$$

Definition 3.13. *For any number $k \in [n+1]$ let $\bar{k} = n - k \in [n+1]$. So that $k + \bar{k} = n$.*

Definition 3.14. *We say that x is **dominated** by $y \in \mathcal{C}$ iff they are adjacent in G_n. Then if $\mathrm{wt}(y) = \mathrm{wt}(x) + 1$ we say that x is **postdominated** and if $\mathrm{wt}(y) = \mathrm{wt}(x) - 1$ it is **predominated**.*

Observation 3.11. *In a safe code $\mathcal{C}$ any word $x \in G_n$ is either postdominated or predominated, not both nor none.*

Proof. We already know that if $\mathcal{C}$ is safe any word $x \in G_n$ is dominated exactly once. Also any element $x \in G_n$ is only at distance one from elements in the stage below or above it so it can only be dominated from $\mathrm{wt}(x) - 1$ or $\mathrm{wt}(x) + 1$. □

Observation 3.12. *The a_k elements $x \in \mathcal{C}$ in stage k postdominate ka_k elements (in stage $k-1$) and predominate $\bar{k}a_k$ elements (in stage $k+1$) in total.*

Proof. Each x has k neighbours in stage $k-1$ and $\bar{k}$ neighbours in stage $k-1$ corresponding to taking turning each of the k 1's of the vector to 0's or each of the $\bar{k}$ 0's in the vector to 1's, and any element is dominated exactly once. □

This motivates the following definition:

Definition 3.15. *For any $k \in [n+1]$, let $b_k := ka_k$ and $\bar{b}_k := \bar{k}a_k$.*

[13] Which is know from topic of discrete math in HL IB math.

[14] Recall definition 1.5 of *stage*.

Lemma 3.3. *If the code is safe then the following recurrence relation is held:*

$$b_{k+1} + \bar{b}_{k-1} = \binom{n}{k} \tag{22}$$

for $k = 1, 2, \cdots, n$.

Proof. By observation 3.12, in stage k, b_{k+1} elements are postdominated and $\bar{b}_{k-1}$ elements are predominated. Then they must add up to $|I_k^n| = \binom{n}{k}$. □

Observation 3.13. *Since* b *and* $\bar{b}$ *are directly related to* a_k *this recurrence allows to compute* a_k *for each* $k \in [n+1]$ *if we know* a_0 *and* a_1. *We already mentioned in observation 3.4 that* $a_1 = 1$ *similarly* $a_{n-1} = 1$. *On the other hand* a_0 *is* 1 *for the elementary codes and* 0 *otherwise. The recurrence runs independently for the even and odd indices because* $k - 1 \equiv k + 1 \mod 2$. *Thus with* $a_1 = 1$ *we may get* a_k *for all* $k \in [n+1]$ *odd. To compute it for the evens we need to distinguish elementary and non-elementary cases. In any of the cases the sequence* a_k *is determined for each* n.

Case 4. *For* $k \in [n+1]$ *odd.*

Computing some terms for a general n *we conjecture* $a_k = \frac{1}{n}\binom{n}{k} = \frac{(n-1)!}{k!\bar{k}!}$ *so that* $b_k = \binom{n-1}{k-1}$ *and* $\bar{b}_k = \binom{n-1}{k}$. *Which makes a lot of sense: all the codes follow the same distribution so it is necessary to have* $\frac{1}{n}$ *of the total in each stage.*

Proof. Since the sequence is deterministic we only need to check that indeed $a_1 = \frac{(n-1)!}{1!\bar{1}!}\frac{(n-1)!}{(n-1)!} = 1$ and that:

$$b_{k+1} + \bar{b}_{k-1} = \binom{n-1}{k} + \binom{n-1}{k-1} = \binom{n}{k}$$

Which follows from the recursive definition of Pascal's triangle. □

Case 5. *For* $k \in [n+1]$ *even if we try to compute terms in general for the two terms the conjecture seems clear until* $k = 6$ *but then we realize it is false and for* $k = 8$ *seems very nasty. The solution cannot be the same because* $\frac{1}{n}\binom{n}{k} \notin \mathbb{N}$ *for* $k = 0$. *But we can apply what we learned from the previous case and set:*

$$a_k = \frac{1}{n}\binom{n}{k} + a'_k \tag{23}$$

And now find the general solution a'_k *which can be added to the particular solution* $\frac{1}{n}\binom{n}{k}$ *as it is commonly done with differential equations. Now:*

$$\begin{aligned} b'_{k+1} + \bar{b}'_{k-1} &= 0 \\ (k+1)a'_{k+1} = b'_{k+1} &= -\bar{b}'_{k-1} = -(k-1)a'_{k-1} \end{aligned} \tag{24}$$

Setting $a'_0 = c$ *we clearly get* $a'_k = c(-1)^{k/2}\binom{n/2}{k/2}$. *Then* $a_0 = \frac{1}{n}\binom{n}{0} + a'_0 = \frac{1}{n} + c$ *so* $c = a_0 - \frac{1}{n}$ *which is* $\frac{n-1}{n}$ *for elementary codes and* $-\frac{1}{n}$ *for non-elementary. Like this we recover:*

$$a_k = \frac{1}{n}\binom{n}{k} + c(-1)^{k/2}\binom{n/2}{k/2} \tag{25}$$

In summary:

$$a_k = \begin{cases} \frac{1}{n}\Re\left(\binom{n}{k} + (n-1)i^k\binom{n/2}{k/2}\right) & \text{for elementary codes} \\ \frac{1}{n}\Re\left(\binom{n}{k} - i^k\binom{n/2}{k/2}\right) & \text{for good non-elementary codes} \end{cases} \tag{26}$$

where $\Re(x)$ is the real part of x for $x \in \mathbb{C}$ and $i := \sqrt{-1}$ the imaginary unit.

We already know that $a_k \in \mathbb{N}$ for every $k \in [n+1]$ and $n = 2^N$ because the codes exist, but it is a pretty hard problem to give a number theory proof from here. Indeed if it wasn't true then it would have been a safe way to disprove claim 2.1, that's why we considered this approach.

4 Similar smaller problem

Now comes the part where we try to get inspiration from easier problems. Consider the following quite typical but interesting problem:

Problem 2. *Now we have* 100 *prisoners and also a guard that gives them a last chance to die or escape free. This time he explains a different situation:*

—I will put the 100 *of you in a row, each of one wearing a hat which can be of* 2 *colours: black or white. So that you will only be able to see the colour of the hats of the prisoners that you have in front. Then each of you will have to guess the colour of your hat, starting for the last of the row (the one that sees anybody except that no one sees him) and moving in front. If you guess right you live, else you are immediately killed. If someone says something different than "black" or "white" everyone is executed. Now you have some time to argue the strategy.*

What is the best strategy they can play (meaning the one that ensures more people to live).

An easy strategy one can think is just that the odds say the hat of the one in front so evens live. Like this only half live for sure. But there are better strategies. As we did with the other problem here we will follow the plan 0.1 we set in introduction:

4.1 Reformulate

Observation 4.1. *The first prisoner to speak has no chance to certainly know his colour. This tells us to things: first that he may "sacrifice" his choice for the group and say something that gives information to the others, and second that the ideal strategy we are searching for, if there is, is that they all live except maybe the last one with* $\frac{1}{2}$ *probability.*

Observation 4.2. *On the time one has to speak he knows two things:*

1. *The binary choice of the first prisoner to speak.*
2. *The colour of the hat of everyone except him.*

This is because he sees the colour of the hat of everyone in front of him and he knows the ones behind because if they guessed wrong they have been killed. This allows us to see the problem in another way, which seems much easier: we can assume that instead of being placed in a row they are placed in a circle, where everyone sees everyone.

To formalize we say, similarly as before:

- $n = 99$
- Assign number 0 to **white** 1 **black**.
- Number each prisoner from the first in the row: 0 to the last n.

Then we will also take the order of the $n-1$ people (without counting the last one) in the row and see the configuration of 0's and 1's as a binary word or vector with addition, etc. Thus it is an element of our abstract object G_{n-1}. Then we define similarly as before, let $F_n : G_{n-1} \to G_1$ be *the function that the first prisoner plays*, in the sense of the colour that he says given a certain configuration of the other $n-1$ prisoners, be called the **strategy**.

Definition 4.1. *A strategy will be called* ***safe*** *iff it ensures everyone lives except maybe the first one to speak, i.e. iff it allows the other prisoners to know their hats.*

Then we will focus on searching for safe strategies.

4.2 Development

Consider the following important argument:

Observation 4.3. *Suppose* F_n *is a safe strategy with* $n+1$ *prisoners. Then consider a general configuration* $x \in G_n$ *and a neighbour configuration* $x+e_i$*. Then suppose* $F_n(x) = F_n(x+e_i)$*, then the* i*'th prisoner cannot distinguish both situations, which means he cannot guess right in both. This means* $F_n(x) \neq F_n(x+e_i)$ *or* $F_n(x)+1 = F_n(x+e_i)$ *for any* $x \in G_n$ *and* $i \in [n]$ *(always in* $\mod 2$*). This at the end implies* $F_n(x) = \mathrm{wt}(x) + F_n(0)$*.*

Proof. A short proof would be just saying: let $x = (x_0, x_1, \cdots, x_{n-1}) = \sum_{i=0}^{n-1} x_i e_i \in G_n$ then:

$$F_n(x) = F_n\left(\sum_{i=0}^{n-1} x_i e_i\right) = x_0 + F_n\left(\sum_{i=1}^{n-1} x_i e_i\right) = \cdots = \sum_{i=0}^{n-1} x_i + F_n(0) = \mathrm{wt}(x) + F_n(0) \quad (27)$$

Of course there are several other ways to justify it. For example induction on $\mathrm{wt}(x)$ would be more formal. Or also we can consider the set $\ker(F_n) \subseteq G_n$ and observation 4.3 is equivalent to: there are no neighbours in $\ker(F_n)$ or it is a Hamming code forbidding distance 1. Since the graph G_n is bipartite (see: A.3) $\ker(F_n)$ must be subset of one of the parts and $G_n \backslash \ker(F_n)$ also so they are **the** parts, which are $\{x|\mathrm{wt}(x) \equiv 0 \mod 2\}$ and $\{x|\mathrm{wt}(x) \equiv 1 \mod 2\}$. Thus either $F_n = \mathrm{wt}$ or $F_n = \mathrm{wt} + 1$. □

Then the strategy is can be explained in normal words as: the first prisoner counts the parity of the number of black hats and says white or black depending on what they've agreed. Then the others needs to count the parity of the number of black hats of the others and compare it to the total to know that they are white if it is the same or black otherwise.

4.3 Generalization

First possibility is to generalize n. The solution is exactly the same, but as we said, a for example possibility would also have been to proof the bipartitness of G_n by induction.

We can also generalize the number of colours. We can therefore propose the same problem with k colours. Then what we will do is first assign numbers to colours: each of the k colours will have a number assigned from the set $[k]$. We represent the configurations using the vector space $\mathbb{F}_k^{n-1}$. As we've said, there is a good idea that does help in this kind of generalizations: do the same but in the new vector space.

For example consider $F_n(x) = \sum_{i=0}^{n-1} x_i = x1_n$ (where $1_n := \sum_{i=0}^{n-1} \tilde{e}_i$). Then clearly it works because knowing x_j for all $j \in [n] \setminus \{i\}$ and $F_n(x)$, i'th prisoner can recover his own hat x_i.

The main difference is that this in not unique solution in the generalization, there are now several variants we can introduce. For example take a sequence of permutations $\sigma_0, \sigma_1, \cdots, \sigma_{n-1} \in S_k$. Then consider $F_n(x) = \sum_{i=0}^{n-1} \sigma_i(x_i)$ clearly from there we can also recover x_i if we know all the other hats. Although some are the same, this introduces basically $k!^n$ variations.

In Hamming space and graph formulation if G_n^k is the Hamming space with n dimensions using the field $\mathbb{F}_k$ we are searching for a normal colouration (forbidding adjacent vertices with the same colour).

5 Research and Applications

With such a bunch of keywords that we highlighted in section 1, one can easily find several articles such as [4, 6, 7], which talk about similar problems on the Hamming space. For example, we can find a big "spoiler" on this articles:

Definition 5.1. *Let $\chi_k(n)$ be the minimum number a needed to partition the Hamming space into a codes in such a way that any code has two words at distance k (property of forbidden distance k in codes).*

Then the following result is known:

Theorem 5.1 (χ_2 **bound**).

$$n \leq \chi_2(n) \leq 2^{\lceil \log_2 n \rceil} \tag{28}$$

Corollary 5.1.1. *For $n = 2^N$:*

$$n \leq \chi_2(n) \leq 2^{\lceil \log_2 n \rceil} = 2^{\lceil \log_2 2^N \rceil} = 2^{\lceil N \rceil} = 2^N = n \tag{29}$$

Thus $\chi_2(n) = n$ for a power of 2.

Which is exactly claim 2.1.

Also, for example, the idea of linear codes came to my mind just as we explain it in section 3. But we didn't invented it, it's an important concept in coding theory that one has to take always into account (see: [2, 3]) because can solve most of the problems.

A much more general question discussed in coding theory is normally formulated as follows:

Definition 5.2. *The set of distances of a code $\mathcal{C}$ in G_n^k is $D(\mathcal{C}) := \{d(x, y) | x, y \in \mathcal{C}, x \neq y\}$.*

The question is to find the maximum sized codes with $D(\mathcal{C}) \subseteq D$ for a certain D for every $n, k \in \mathbb{N}$. Then in our particular case we were searching for maximum codes with $D = ([n] + 1) \setminus \{2\}$. Normally the most interesting question for its applications is the case $D = [n + 1] \setminus [d]$ for a certain d, i.e. codes with minimum distance $d + 1$. This is because this codes allow us to auto-correct up to $\lfloor d/2 \rfloor$ errors. For example consider the following problem:

Problem 3. — *I'm thinking of a number from the set $[n]$. You have to guess which is he number using questions of* yes *or* no*. But alert! You will first have to say all the questions and then I will answer them all lying in exactly* 1 *of them.*

What's the minimum number of question you need to ask to be able to guess it for sure?

Now that we already know the typical procedure to solve kind of problems we will go faster:

- **Yes** is 0, **No** is 1.
- A question is seen as a binary vector from $q \in G_n$ setting i'th entry to 0 or 1 depending on the the answer to that question if the number was i.

Observation 5.1. *If the set of answers to a question is the same the question is equivalent and thus there are only $|G_n| = 2^n$ fundamentally different questions.*

Definition 5.3. *A **query** is a sequence of questions $q_0, q_1, \cdots, q_k \in G_n$ and is seen as a matrix $Q = \sum_{i=0}^{n-1} q_i \tilde{e}_i$. Then call the "hypothetical answers" of Q to the columns: $p_0, p_1, \cdots, p_{n-1} \in G_k$ i.e. $p_i := (Q\tilde{e}_i)^T$. Then the code $\mathcal{C}$ of a query Q is defined as the set of hypothetical answers: $\{p_i | i \in [n]\}$.*

Definition 5.4. *A query Q is called **safe** if it allows us to guess the number with the rules of the game.*

Definition 5.5. *An **answer** to a query Q is the sequence of answers to each question, again it is seen as a vector $a \in G_k$ (being k the number of questions).*

Observation 5.2. *If my number is i and I answer a to your query Q. Then $d(p_i, a) = 1$ because the sequence of answers must be the correct answers for i except for one.*

Observation 5.3. *A query Q is safe iff there are no p_i, p_j with $d(p_i, p_j) = 2$.*

Proof. If there is so, then it could happen that the answer was an a such that $d(p_i, a) = d(p_j, a) = 1$ and then you won't know if the number is i or j. Conversely if this doesn't happen then there is no possibility that this fastidious situation happens and then there will be no more that 1 p_i at distance 1 from a. Also there cannot be 0 elements at distance 1 because then I would be giving the information wrong or not thinking of a concrete number. □

Observation 5.4. *Now the problem is equivalent to finding the minimum k for which G_k has a code $\mathcal{C}$ of size n with property of forbidden distance 2. Once we find such a code $\mathcal{C}$ we just have to put the codewords in some order as columns $p_i \in \mathcal{C}$ on a matrix Q to construct the query, from which we can recover the set of questions q_i as the rows in some order. Then just formulate the question q_i in the form: "Is the number from the set $\{j \in [n] | q_i \tilde{e}_j = 0\}$?"*

Clearly it es pretty much equivalent to our first problem.

Of course the question may be generalized saying that I can lie in e questions for any e in a set $E \subseteq [n+1]$. Then is when we need to find a code $\mathcal{C}$ with $D(\mathcal{C}) \subseteq D$ (particularly for $D = [n+1] \setminus (E + E)$) if possible.

The applications of this problem to engineering are clear when *I*'m replaced by any *transmitter*, and *you* are replaced by a *receptor*, say at the other side of the galaxy. Then I'm not intentionally lying to you but there is some *noise* that can alter some of the bits of the message causing some error. Then we basically have a probability $p < \frac{1}{2}$ of getting an error on a bit so that we want to maximize the probability that the message is still auto-corrected but at the same time minimize the size of the message.

The common languages we speak automatically do this, they are redundant so that even thought we miss a letter we can figure out the content of the message. Coding theory just tries to do this but in a controlled way so as to be able to have the most efficient communication or encoding possible.

Let's put some names:[15]

Definition 5.6. *The message that the transmitter sends will be called $M \in \mathcal{C}$.*

Definition 5.7. *The message the receptor receives will be called $R \in G_n^k$.*

Observation 5.5. *The number of errors in bits $d(R, M)$ in a message follows a binomial distribution $d(R, M) \sim B(n, p)$.*

Definition 5.8. *The decoding function $\mathcal{D} : G_n^k \to \mathcal{C}$ translates $R \in G_n^k$ received into $M \in \mathcal{C}$.*

Observation 5.6. *The most intuitive decoding function to use is the one that assumes minimal error: $\mathcal{D}(R) = M$ being $M \in \mathcal{C}$ the one minimizing $d(R, M)$. And some random if there are more than one at minimum distance. Then what makes sense is to have $D(\mathcal{C}) = [n+1] \setminus [d]$ for the maximum possible d, with which one will surely guess right with a probability:*

$$\Pr\left(d(R, M) \leq \lfloor d/2 \rfloor\right) = \sum_{i=0}^{\lfloor d/2 \rfloor} \binom{n}{i} p^i (1-p)^{n-i} \tag{30}$$

[15] All this can be found on [3].

6 Conclusion

We got to prove our claim using only two quite intuitive brilliant ideas. First, know that we need to find codes. Once we have a code we suggest that a normal configuration can be applied to get a safe strategy, even though we don't prove if it is the only way to do it. Finally, to find safe codes we apply the other small intuitive assumption: we may search for linear codes. Once we have imposed a normal configuration using linear codes it suffices to do some *routine* work to find all possible safe strategies.

Also we show some ways in which any safe strategy can be varied and hence we prove some results about the number of solutions and safe codes. In the generalization where we study the first smaller cases we also saw some things that help us conjecture or disproof conjectures.

Finally we underline other approaches that may be complementary or provide another way to attack the problem such as the linear algebraic approach and stages analysis.[16]

As opened questions we leave for further researches:

Question 2. *Are there two safe disjoint codes $\mathcal{C}_1$ and $\mathcal{C}_2$ such that there is no $a \in I_2^n$ with $\mathcal{C}_2 = a\mathcal{C}_1$.*

Hence or otherwise:

Question 3. *Is conjecture 3.1 true?*

Plus:

Question 4. *What's exactly C_n or particularly $|C_n|$?*

Hence from questions 3 and 4 or otherwise:

Question 5. *How many safe strategies are there for each $n = 2^N$?*

We know from our research that this isn't known yet. The best we can find approaching question 4 is [5], but questions 2 and 3 seam an easier way to start.

Finally in all coding theory branch the questions are asked for more general properties of the codes other than safeness as explained in section 5.

[16] See sections 3.3 and 3.4, respectively.

A Some theory

A.1 To understand our working field

A.1.1 Groups

In algebra and group theory, a *group* $(G, +)$ is just a set of elements G (we may call them numbers) equipped with an operation $+$ (we may call it addition and in our case is indeed addition). This operation does not need to be commutative[17] (but in our case it is and then is called Abelian or commutative group) but needs to be associative[18]. Also a group needs to be equipped with an identity (we may call it $0 \in G$, and in our case is the 0) which *does nothing* when added to other elements ($a + 0 = a$), and an inverse $-a \in G$ for any element $a \in G$, which is the one that does $a + (-a) = 0$.

When a group is finite (in our case it is) we may represent it with a table indicating the sum of each pair of elements in the group (which must also be a member of the group)[19]. For example, we see the table of the group $\mathbb{Z}_2$ below:

$+$	0	1
0	0	1
1	1	0

Table 4: Cayley table for addition on group $\mathbb{Z}_2$

As one can check it is indeed associative, 0 acts as identity and the inverse of 0 is 0 ($0 + 0 = 0$), while the inverse of 1 is 1 ($1 + 1 = 0$). Also, one can observe that it is commutative and, rare enough, what I call *self-inverse* because the inverse of any element is itself.

We say that this is the group of the *integers modulo* 2, and thus write $\mathbb{Z}_2 := \mathbb{Z}/2\mathbb{Z}$ basically because 0 and 1 are the possible residues modulo 2 and we add them like seen in the table above[20].

Similarly we can talk about the integers modulo k which induce the group $\mathbb{Z}_k = \mathbb{Z}/k\mathbb{Z}$.These are also called: cyclic groups.

A.1.2 Rings and fields

When we add a new operation to a commutative group (this time lets call it multiplication $(\cdot)$), also with its associativity and identity element (call it 1, and in our case is indeed 1), we form a *ring* $(G, +, \cdot)$. But this is only is allowed when multiplication and addition satisfy distributive property (left and right)[21].

Furthermore, when there exist also multiplicative inverses, i.e. $a^{-1} \in G$ such that $a \cdot a^{-1} = 1$ for each element $0 \neq a \in G$ then it can be called a *field*[22]. This is what happens to our group $\mathbb{Z}_2$ with the multiplication table seen below:

$\cdot$	0	1
0	0	0
1	0	1

Table 5: Cayley table for product on field $\mathbb{Z}_2$.

As one can check distributive property holds, 1 is the product identity and 1, of course, has a multiplicative inverse (also 1). The product is also commutative (so it is a commutative field).

[17] Commutative property is $a + b = b + a$.

[18] Associative property is $(a + b) + c = a + (b + c)$

[19] All this is known theory from option *Sets, Relations, and Groups*, of HL IB math.

[20] Modular arithmetic is known from option *discrete math*, of Math HL IB. If you don't know how to add modulo k is just normal addition and then subtract k if the result is greater than $k - 1$.

[21] This is $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(b + c) \cdot a = b \cdot a + c \cdot a$.

[22] By distributive property and definition of identity sum $a \cdot b = a \cdot (b + 0) = a \cdot b + a \cdot 0 \implies a \cdot 0 = 0$ unless 0 is the only element of the group $a \cdot 0 \neq a$ for $a \neq 0$, thus $0 \neq 1$ and then $a \cdot 0 = 0 \cdot a = 0 \neq 1$ so 0 cannot have a multiplicative inverse.

As before this also happens, as one can guess, with $\mathbb{Z}_k$, but this time its only a ring not a field when k is composite (just because the divisors of k have no multiplicative inverse[23], as $0 \equiv k$). Other easy examples of infinite rings are the integers $\mathbb{Z}$, and fields: the rationals $\mathbb{Q}$, the reals $\mathbb{R}$ and the complexes $\mathbb{C}$.

It's possible that the ones more familiar with computer science or electronics see this table and think about the logic operations or logic gates. Indeed $(+)$ plays exactly as (XOR) and $(\cdot)$ as (AND) and I admit I first thought it like this. For the moment, I prefer working with plus and dot signs, and there is some reason we will see later.

A.1.3 Vector spaces

There exist also vector spaces $(V, \mathbb{F})$. These are group of elements V (this time we call them vectors, and in our case will be the n-dimensional binary vectors: $\{0,1\}^n$) within a field $\mathbb{F}$ (in our case $\mathbb{F} = \mathbb{Z}_2$). As a group, V is equipped with and operation of sum of vectors $+$, which must be again associative, etc. and as a vector space with a scalar product $(\cdot)$ that allows us to multiply vectors from our set V by numbers from our field $\mathbb{F}$ to give other vectors from the set V. In our case we are considering the vector space of the binary n-dimensional vectors $\mathbb{Z}_2^n$ where the sum is performed modulo 2 componentwise and the product is just $1 \cdot v = v$ and $0 \cdot v = 0$).

A.1.4 Generator and basis

A generator of a group is just a subset of elements of the group, which, when combined all together with the operation they can form any element of the group. For example, the set $\{1\}$ is a generator of the group $\mathbb{Z}_k$ because one can add 1 to itself i times to get i and $k \equiv 0$.

Similarly for a vector space, a basis is a generator of the set of vectors with the sum and the scalar product. For example in the classical vector space $\mathbb{R}^k$ (with the field $\mathbb{R}$), the set of vectors $B := \{e_1, e_2, \cdots, e_k\}$ where $e_i := \underbrace{(0, 0, \cdots, 0, 1, 0, \cdots, 0)}_{1 \text{ in the } i\text{'th position}}$ is a basis because one can express any vector $v = (a_1, a_2, \cdots, a_k)$ using the rules of sum and scalar product as $v = a_1 e_1 + a_2 e_2 + \cdots + a_k e_k$. The basis B defined in the same way can be used as generator of the group $\mathbb{Z}_p^n$ or similarly a basis of the vector space $\mathbb{Z}_p^n$ with the field $\mathbb{Z}_p$ in general for p prime and in particular for $p = 2$. [24]

A.2 Graphs and the hypercube graph

A simple graph $G = (V, E)$ is just a set of elements V called vertices or nodes and a set of edges $E \subseteq \binom{V}{2}$, which are pairs of vertices[25]. When there is an edge going from vertex u to v we say that they are connected or adjacent[26].

Now it reminds us to investigate how exactly does our graph "look" like.

Observation A.1. *The easiest way to visualize it is taking an n-hypercube, which is a generalization of segment, square, cube, in n dimensions. Then the vertices are the vertices and the edges are "edges" or rather "hyperedges". That's why is called the hypercube graph. This is equivalence is because of the vector space interpretation, seeing the vectors as points in a Cartesian system, two points are connected if only a coordinate changes.*

There is also an inductive definition of G_n which might help:

Definition A.1. *Let G_n be defined recursively as:*

$$G_k := \begin{cases} K_1 & \text{for } k = 0 \\ G_{k-1} \times K_2 & \text{for } k > 0 \end{cases}$$

Where K_k is the k-complete graph, i.e. K_1 is a lonely vertex and K_2 two vertices joined together.

[23] As we see again in discrete math option there are linear Diophantine equations with no solution due to divisibility, it's the same.

[24] For more information on this section consult [9, 10, 3].

[25] The notation $\binom{S}{a}$ is used as the set of subsets of a elements of S

[26] This is known from option *discrete math* in HL IB math.

Observation A.2. *With the previous definitions G_n also corresponds or is isomorphic to the hypercube graph. Thus we can also say $G_n = K_2^n$. Intuitively, $G \times K_2$ is constructed duplicating G and connecting each vertex with its correspondent in the copy, so doing this n times is a way to construct G_n. Also then notice that $G_n \times G_n = G_{2n}$.*

Another important fact of the hypercube graph:

Observation A.3. *Consider the set of vertices that have and odd number of ones and the ones that have an even number of ones. Clearly an edge goes from a vertex with odd to one with even or the other way around. Thus G_n is bipartite (for any $n \in \mathbb{N}$) using the parts named before.*

[27]

A.3 Coding theory

Coding theory is the study of the properties of codes. A code is basically like a dictionary $\mathcal{C} = \{w_1 \to v_1, w_2 \to v_2, \cdots, w_k \to v_k\}$ or a function $\mathcal{C} : A^* \to B^*$ mapping words w_1, w_2, ..., w_k formed from an alphabet A, to words v_1, v_2, ..., v_k formed from an alphabet B. Normally the most studied alphabet is again $\{0, 1\}$ for its applications on computer science, electronics, communications, etc.

Throughout our project we are only interested in one side of the code, the set of binary words or the domain. Thus for us, a code will be just a set of binary words of the same length, i.e. a subset of $\{0, 1\}^n$.

A.3.1 Metric spaces and Hamming space

A space is basically a set of points P. There are several types of spaces, but when there is a function $d : P^2 \to \mathbb{R}$ indicating the *distance* $d(a, b)$ between two points $a, b \in P$ we call it a *metric space*.

One instance of metric space, very useful and precisely created in coding theory for our problem, is the so called *Hamming space* G_n. The Hamming space has, as set of points, guess what, the binary words $\{0, 1\}^n$ though it can also be defined in any other vector space like Q^n. Two functions are defined there:

Definition A.2. *The Hamming weight* $\mathrm{wt} : Q^n \to \mathbb{Z}$ *is the number* $\mathrm{wt}(w)$ *of* 1*'s in a word* w*, or more generally when it is not binary the number of non-zeroes.*

Definition A.3. *The Hamming distance* $d : Q^n \times Q^n \to \mathbb{Z}$ *can be defined as* $d(a, b) := \mathrm{wt}(a - b)$ *is the number of positions, letters or entries (whatever we want to call it) that one have to change to get from word* a *to* b*.*

Observation A.4. *The distance function can be seen as the error $d(a, b)$ of a message a when instead b was send, this way is where the applications on engineering comes from*[28]*.*

Observation A.5. *As in the euclidean space, this metric obeys the triangular inequality: $\forall x, y, z \in Q^n$:*

$$d(x, z) \leq d(x, y) + d(y, z) \tag{31}$$

even though for example if we say that $x, y, z \in Q^n$ are aligned iff the equality case in 31 holds, then x, y, z aligned and x, w, z aligned do not imply y, z, w aligned as in the euclidean space.

Observation A.6. *Another important observation about the Hamming distance function is that:*

$$d(x, y) \equiv \mathrm{wt}(x) - \mathrm{wt}(y) \mod 2 \tag{32}$$

[29]

[27] For more information on this section consult [11].
[28] See section 5
[29] For more information on this section consult [3].

A.4 Direct or Cartesian product

The Cartesian product $A \times B$ of two sets A and B is defined as $A \times B := \{(a, b) | a \in A, b \in B\}$, i.e. the set of pairs of elements of A and B. Direct product is a generalization of the Cartesian product that can also be applied to groups, graphs, functions, etc., so I'll refer to both indistinctly with the notation $\times$.

The Cartesian product of two groups $G = (V, +)$ and $H = (W, \cdot)$ is the new group $G \times H = (V \times W, *)$ formed by the set of elements $V \times W$ with operation $*$ defined as the componentwise operation $+$ and $\cdot$, i.e. for two elements $a = (v, w)$ and $b = (u, x)$ of $V \times W$, $a * b = (v + u, w \cdot x)$. The inverse of a pair is the pair of the inverses and the new identity element is the pair of the identities.[30]

The Cartesian product of two graphs $G = (V, E)$, and $H = (W, F)$ is the new graph $G \times H = (V \times W, E \otimes F)$ [31] formed by the set of vertices $V \times W$ and an edges $E \otimes F := \{((v, w), (u, x)) \in |(v, u) \in E, (w, x) \in F\}$, i.e. an edge going from (v, w) to (u, x) if v and u were adjacent in G and w and x were adjacent in H. [32]

In any of the interpretations of Cartesian product seen above, the Cartesian power is $A^k := \underbrace{A \times A \times \cdots \times A}_{k\ A\text{'s}}$ for a natural number k, i.e. A^k represents the set of k-tuples of elements of A or with its operations, edges or whichever properties. [33]

A.5 Results

Theorem A.1 (**Fundamental theorem of Abelian groups**). *Any finite Abelian group G is isomorph to:*

$$\bigtimes_{i=1}^{k} \mathbb{Z}_{a_i} \tag{33}$$

for some $a_1, a_2, \cdots, a_k$, where $\bigtimes$ represents a Cartesian product, or in this case direct sum of the groups.

Proof. We outline a proof by strong induction on $|G| \in \mathbb{Z}^+$. The base case $|G| = 1$ is trivial $G = \{0\} \cong \mathbb{Z}_1$. For the induction step take an $a \in G$ and consider the subgroup $[a]$ generated by a. The check that the quotient $G/[a]$ satisfies $G \cong [a] \times G/[a]$, that $[a] \cong \mathbb{Z}_n$ (being n the order of a and that $|G/[a]| = |G|/|[a]| = |G|/n < |G|$ thus by hypothesis $G/[a]$ is isomorph to a product as in 33 and then $G \cong \mathbb{Z}_n \times G/[a]$ which is of the same form. □

Still to go over:

- Homomorphisms
- Cosets, subgroups and quotient group
- Fundamental theorem of Abelian groups

[30] This operation of a combination of groups is also called direct product in group theory or direct sum ($\oplus$) when working with Abelian groups.

[31] This notation of $\otimes$ is just a similar variation of Cartesian product of sets, just my invention.

[32] After all the similarities we have seen, one could expect to find also the Cartesian product of spaces. Of course one can also define a new space $S_1 \times S_2$ from two spaces S_1 and S_2 with sets of points P_1 and P_2, respectively using the new set of points $P_1 \times P_2$. The problem comes with metric, since there is no generally accepted way to define the new metric d, for example the Hamming space viewed as a Cartesian power of a space would use the rule $d = d_1 + d_2$ whereas the Euclidean k-dimensional space would be using the rule $d^2 = d_1^2 + d_2^2$ to define the new distance.

[33] For more information on this section consult [1].

B Computer search

B.1 Algorithm

This is an explanation of the general idea of the algorithm to find safe codes:

- Have a set of elements which will be my code (variable "Code").
- Each time, add a codeword that **can** added maintaining the property of forbidden distance 2 (so from the set "available") until one can't add more codewords.
- To decide which codeword to add each time we use property of dominating and we are actualizing a set of elements (*tofulfill*), which are those that are not yet *dominated*. Then we chose an available neighbour of some element tofulfill.
- Either what one has at the end is safe or else is not sufficiently large.
- To find them all we are doing a backtracking (which increases a lot the complexity of the process).
- To not repeat codes chosen in different order, erase all the ones that we did not choose from the available.

B.2 Possible upgrades

- Take the elements stage by stage.
- We are only searching for elementary codes, and as we said in section 3.1 we could further restrict it to primitive codes.
- See [5], which, as stated in section 3.1.2 develops very powerful tools to do an efficient search.

This program executes case $n = 8$ in half-hour approximately so we would need a much more important improvement of complexity to solve case $n = 16$ in a reasonable time. Maybe with some effort, a supercomputer can do it.

B.3 Codes and outputs

The code and the outputs can be found at https://github.com/PauCantos/Coding-theory-search-algorithms with the names "search_codes2.cpp" and "outputs/search_codes2(3).tex", respectively.

You can also find other interesting things like "adjacency_matrix.cpp" and "stages_analysis(c).cpp" from stages analysis (see section 3.4) or compare other programs.

B.4 Implications of the output

As we've said there are 7200 elementary codes, (hence 900 of which are primitive), see case 3 in section 2.1.1.

From this output we shall, for example, take the second code:

$$\begin{array}{{cccccccc}}
\{00000000, & 01010100, & 10010101, & 11000010, & 00110010, & 01100101, & 10100100, & 11110000, \\
00000001, & 01010110, & 10010111, & 11000011, & 00110011, & 01100111, & 10100110, & 11110001, \\
00001110, & 01011001, & 10011000, & 11001100, & 00111100, & 01101000, & 10101001, & 11111110, \\
00001111, & 01011011, & 10011010, & 11001101, & 00111101, & 01101010, & 10101011, & 11111111\}
\end{array}$$

which at first seems linear until one reaches 9'th and 10'th codewords with 01010100 and 01010110 respectively, that have difference 00000010, which is not in the code. Like this one, most of them are not linear so this is the disproof by example of 3.2.

References

[1] Wikipedia the free encyclopedia. *Cartesian product*. URL: https://en.wikipedia.org/wiki/Cartesian_product.

[2] Wikipedia the free encyclopedia. *Linear code*. URL: https://en.wikipedia.org/wiki/Linear_code.

[3] Conrado Martínez Gadiel Seroussi. *Introduction to Coding Theory*. UPC. Aug. URL: https://www.cs.upc.edu/~conrado/docencia/codigos.pdf.

[4] Dongsoo S. Kim, Ding-Zhu Du, and Panos M. Pardalos. "A coloring problem on the n-cube". In: *Discrete Applied Mathematics* 103.1 (2000), pp. 307–311. ISSN: 0166-218X. DOI: https://doi.org/10.1016/S0166-218X(99)00249-8. URL: http://www.sciencedirect.com/science/article/pii/S0166218X99002498.

[5] P. Ostergard and O. Pottonen. "The Perfect Binary One-Error-Correcting Codes of Length 15: Part I—Classification". In: *IEEE Transactions on Information Theory* 55.10 (Oct. 2009), pp. 4657–4660. DOI: 10.1109/tit.2009.2027525. URL: https://doi.org/10.1109/tit.2009.2027525.

[6] Patric R.J. ÖstergÅrd. "On a hypercube coloring problem". In: *Journal of Combinatorial Theory, Series A* 108.2 (2004), pp. 199–204. ISSN: 0097-3165. DOI: https://doi.org/10.1016/j.jcta.2004.06.010. URL: http://www.sciencedirect.com/science/article/pii/S0097316504000937.

[7] Patric R.J. Östergård. "A Coloring Problem in Hamming Spaces". In: *European Journal of Combinatorics* 18.3 (1997), pp. 303–309. ISSN: 0195-6698. DOI: https://doi.org/10.1006/eujc.1996.0093. URL: http://www.sciencedirect.com/science/article/pii/S0195669896900931.

[8] G. POLYA. *How to Solve It*. Princeton University Press, Oct. 2014. DOI: 10.2307/j.ctvc773pk. URL: https://doi.org/10.2307/j.ctvc773pk.

[9] H. A. Priestley and E. Victor Fynn. *Introduction to Groups, Rings and Fields*. Oxford University. URL: https://people.maths.ox.ac.uk/flynn/genus2/sheets0405/grfnotes1011.pdf.

[10] What is a Vector Space? *Introduction to Groups, Rings and Fields*. URL: https://people.maths.ox.ac.uk/flynn/genus2/sheets0405/grfnotes1011.pdf.

[11] Eric W. Weisstein. *Hypercube Graph*. MathWorld–A Wolfram Web Resource. URL: http://mathworld.wolfram.com/HypercubeGraph.html.

```
File: main.tex
Encoding: utf8
Sum count: 9818
Words in text: 8364
Words in headers: 107
Words outside text (captions, etc.): 416
Number of headers: 41
Number of floats/tables/figures: 7
Number of math inlines: 887
Number of math displayed: 44
Subcounts:
  text+headers+captions (#headers/#floats/#inlines/#displayed)
  3+8+0 (1/0/0/0) _top_
  0+1+0 (1/0/0/0) Section: Introduction
  590+3+0 (1/0/0/0) Subsection: Problem and Approach
  68+1+2 (1/1/12/0) Subsection: Notation
  444+1+24 (1/1/18/0) Section: Reformulation}    \label{reformulation
  159+2+11 (1/0/26/2) Subsection: Algebraic formulation
  191+3+12 (1/0/20/1) Subsection: Graph theory formulation
  346+3+0 (1/0/32/0) Subsection: Coding theory formulation
  16+1+0 (1/0/0/0) Section: Generalization
  346+6+8 (2/2/60/2) Subsection: Size of board}  \label{generalization: size of board
  203+2+2 (1/0/11/0) Subsection: Different coin}    \label{generalization: states of coi
  0+1+0 (1/0/0/0) Section: Development}   \label{development
  633+12+0 (3/0/90/12) Subsection: Trivial transformations}    \label{variations
  716+2+30 (1/1/109/8) Subsection: Linear codes
  189+4+12 (1/0/27/3) Subsection: Totally linear algebraic approach}  \label{algebraic a
  499+2+4 (1/0/93/7) Subsection: Stages analysis}    \label{stages analysis
  223+3+0 (1/0/3/0) Section: Similar smaller problem}   \label{section: small problem
  297+1+0 (1/0/12/0) Subsection: Reformulate
  197+1+0 (1/0/24/1) Subsection: Development
  206+1+0 (1/0/20/0) Subsection: Generalization
  898+3+6 (1/0/90/3) Section: Research and Applications}  \label{applications
  263+1+4 (1/0/7/0) Section: Conclusion}    \label{conclusion
  0+2+0 (1/0/0/0) Section: Some theory
  712+14+124 (5/2/98/0) Subsection: To understand our working field}    \label{group the
  254+5+30 (1/0/25/1) Subsection: Graphs and the hypercube graph}   \label{graph theory
  304+7+9 (2/0/35/2) Subsection: Coding theory
  171+4+138 (1/0/48/0) Subsection: Direct or Cartesian product}   \label{cartesian produ
  99+1+0 (1/0/17/1) Subsection: Results
  0+2+0 (1/0/0/0) Section: Computer search}    \label{computer search
  143+1+0 (1/0/1/0) Subsection: Algorithm
  76+2+0 (1/0/2/0) Subsection: Possible upgrades
  49+3+0 (1/0/0/0) Subsection: Codes and outputs
  69+4+0 (1/0/7/1) Subsection: Implications of the output


File: output.bbl
Encoding: utf8
Sum count: 1003
Words in text: 1002
Words in headers: 0
Words outside text (captions, etc.): 0
Number of headers: 0
Number of floats/tables/figures: 0
Number of math inlines: 1
Number of math displayed: 0

Total
Sum count: 10821
```

```
Words in text: 9366
Words in headers: 107
Words outside text (captions, etc.): 416
Number of headers: 41
Number of floats/tables/figures: 7
Number of math inlines: 888
Number of math displayed: 44
Files: 2
Subcounts:
  text+headers+captions (#headers/#floats/#inlines/#displayed)
  8364+107+416 (41/7/887/44) File(s) total: main.tex
  1002+0+0 (0/0/1/0) File(s) total: output.bbl

(errors:27)
```