Bottom up Computing and Discrete Mathematics

P P Martin

March 17, 2011

Contents

Chapter 1

Introduction

1.1 Overview

The idea of this course is to touch on as many aspects as possible of the process of using the power of ‘information’, and of computers, and the theory of discrete mathematics which underlies them.

The aim is to make the student of this course one of the people in their workplace (wherever that might eventually be) who ‘knows about computers’.

The ‘power of computers’ is a very big subject. It includes the things we are aware of computers helping us with, such as

- word processing,
- bookkeeping,
- audio-visual applications and
- design;

as well as more subtle things like

- information gathering
- information organisation,
- information enrichment
- systems simulation,
- optimisation problems,
- robotics

and a million other things.

Discrete Mathematics is the theory underlying all these tools.

Obviously one cannot know everything. Even ‘about computers’. The idea here is to deal with the following problem (which is ‘universal’ in the sense that, if you can solve this one you can work out how to solve all the others!):

Your boss wants a report on XXX[1] by the end of next week. All the necessary information is out there on the web somewhere. But all you have on your desk is a couple of broken computers and some components, and the technicians were laid off in the last round of cutbacks. (And anyway, Google is down for two weeks for legal reasons.) What could you do?

[1]Perhaps XXX is: the price of pork bellies on the Chicago futures exchange?

The course contains two overall components, corresponding to the hardware and 'software' aspects of this problem. The connection between them will be obvious, but they are very different, and can be studied in any order.

The theory part of the course is largely covered in these notes. A suitably concrete version of the problem described above is set up in Chapter 2, and we will start there. As we work through this, we encounter various problems. The theory needed to resolve these problems is contained in the core of the notes, for example in Chapter 5. We will jump there as soon as appropriate, jumping back once we have enough theory to progress our problem, and so on.
There is also a section of more basic Mathematics which we can refer to as necessary, but will not, by default, discuss in class.

The practical part of the course will consist largely of ‘lab’ sessions which we can insert at any time to break up the ‘monotony’ of the theory!

Essentially we will be stripping down a computer to the smallest realistic components, identifying them, understanding them (as far as possible) and then reassembling. (This process works as a metaphor for the operating system as well as the real hardware.) There will only be limited notes on the practicals, and they will not be heavily examined (or otherwise assessed). The motivation for this part of the course will be its very obvious usefulness!

1.2 Prerequisites

We assume that the reader has a knowledge of naive set theory, such as discussed in first year undergraduate ALGEBRA. However we include a brief refresher on this subject in Chapter 4.

1.3 Other support material

- Check out the course homepage.
- Check out the vast amount of related material on the web. Examples:

  `http://en.wikipedia.org/wiki/Discrete_mathematics`

- Check out past exam papers and solutions.
- Check out books called 'DISCRETE MATHEMATICS' (or similar), such as Mattson [**?**].
- Check out Stephenson [**?**], Spiegel [**?**], Ayers [**?**], and other related volumes in the Schaum Outline series.

Chapter 2

Paradigms: Searching

We do not propose to deal with every aspect of the use of computing and discrete mathematics. Rather we will look at examples. We will look at examples that are so profound that, once we have understood them, we will see how to *transfer* this knowledge to all other appropriate problems.

2.1 Searching

Each of us has access to an accumulated store of knowledge and information. This store is only useful in so far as information can be retrieved from it. Methods for retrieving information from the store are, in this sense, as useful as the store itself. The study of *searching* is the study of these methods.

2.1.1 The PageRank Algorithm: searching the web

The web is part of our accumulated store of knowledge and information. *This* store is only useful in so far as information can be retrieved from it. Methods for retrieving information from this store are, in this sense, as useful as the store itself. We propose to study these methods.

The web provides a useful paradigm, in that it is part of cyberspace, and hence accessible by computer (so some of the practical/physical challenges of information retrieval are reduced to a standard form).

Problem: Your boss asks you to prepare a report on XXX. (This is not a school exercise or test of your originality — She doesn't care how you get it, she just needs the answer.) The requirement is to access the knowledge store and mine the relevant information. The simplest way is to see what is on the web.

The difficulty is that there are over 3×10^9 pages on the web. Thus even when you use a search engine to restrict to pages including the phrase XXX (an interesting exercise in its own right, but not one which concerns us for the moment) there will be too many pages — too much information.

What we need is a search engine which will rank all the hits in order, so that the ones most likely to be useful will be ranked at the top. Of course current search engines cannot know what the individual regards as useful! So they have to use a generic algorithm to make this judgement. How could we even begin to formulate such an algorithm?

We need to start with some kind of model or representation of the web. A model which attempted to carry any of the real human meaning of the pages would be (interesting but) extremely complicated. Rather let us begin by thinking of the pages simply as points in cyberspace, and concentrate on the links between them. Thus we have a set of pages, and a set of pairs of pages (i.e. pairs with the property that there is a link between them). Abstractly such a contruct is called a *graph*.

The notion of graph is fundamental in discrete mathematics. If you know all about graphs, continue. If you do not, it is time to jump to the section discussing graphs. Return here when you are done.

Exercise 1 *Go to Chapter 5 and return when you are done.*

We will base our answer on the search engine Google's PageRank algorithm [1].

The idea is to represent the web as a directed graph. The direction of the edges in the graph is from linking page to linked page. For simplicity we will assume that there are no pages without outgoing links.

The PageRank algorithm is not too hard to describe in these terms. It has a single free input parameter p, which should lie in $0 < p < 1$ (we will interpret this later). The other input is the adjacency matrix of the web (!), W say. For simplicity we will consider that $W_{ij} = 1$ if there is a link from page i to page j, and is zero otherwise.

[1] L Page, et al, Stanford University 1998.

The algorithm is run iteratively, at each iteration assigning a value to each page. We will arrange the entire collection of values into a huge vector v (i.e., there is one such for each iteration). To start the process we must assign some initial value to each page (we will later consider the effect of this choice on the final result, for now we will simply assign every page the same initial value). The objective is to end up with a final vector consisting of one number for each web page, where these numbers give the importance rank of the pages (the bigger the number, the more important the page).

The idea behind the algorithm is that the importance of a page is judged on the incoming links. Roughly speaking, if a page is important, people will link their pages to it. More subtly, if it is important, other important pages will link to it. So essentially a page gets a vote from each linking page, but high ranked pages votes count for more.

The question is, how to implement this. Since importance depends on the importance of neighbours (whose importance may well depend on that of the original page), there is a question of existence and uniqueness of solution, not to mention convergence of any concrete algorithm.

Another way of thinking about it which helps with this is to imagine someone browsing the web ‘forever’. Which ever page they are on at time t, they step to one of the linked pages at time $t+1$. Now run this process for a hugely long time, and ask: over all that time, what is the total amount of time spent at each page (adding up all visits). The more important the page, the more often it will be visited. Although other outcomes can be imagined, you will see that it is possible that the *proportion* of the total time of the experiment spent at any given page may eventually settle down to a steady figure. Again the question is, from this idea, how do we get to an actual numerical ranking?

Let v^n be the vector of PageRank values for each web page, at the n^{th} iteration, so that v_i^n is the value for page i. The iteration is

$$v_i^n = (1-p) + p \sum_{j \in \text{pages}} \frac{W_{ji} v_j^{n-1}}{\sum_{k \in \text{pages}} W_{jk}}.$$

Note that for each page j which links to i we have W_{ji} nonzero, so that this page makes a contribution to v_i^n, with the size of this contribution determined by the ranking of j itself at the previous iteration. (The denominator scales down the weight of these provisional ranked votes by the number of outgoing links the voting page has. The idea of this is so that, overall, it casts a total weight of votes, across all linked pages, proportional to its provisional rank.)

If and when the algorithm settles down (i.e. stops changing the values between successive iterations), we may say that it has assigned each rank based on the actual rank of that page's voters.

A number of questions arise. In particular:
How can we implement this algorithm in practice?
How can we tell if the algorithm settles down?
To answer these questions we can usefully recast our problem in the formalism of Stochatic Processes.

2.2 Stochastic Processes

Let M be a matrix, v a column vector, and $()^t$ the transpose operation. Consider the elementary fact of matrix multiplication

$$Mv = v' \qquad \Rightarrow \qquad v^t M^t = (v')^t$$

This 'duality' implies that M and M^t contain essentially the same information, and any property of the rows of M can be states as a property of the columns of M^t. If we wish to manipulate equations of the form above, the use of M or M^t depends simply on whether we prefer to use row or column vectors. Of course having made a choice it is necessary to be consistent within any given calculation.

A matrix may be called *(column) stochastic* if its column sums are all 1, and all the entries are probabilities (i.e. non-negative). [2]
A matrix is called row stochastic if it is the transpose of a column stochastic matrix.
Different authors use different conventions as to whether the term *stochastic matrix* means row or column stochastic. As noted above, the difference is simply a matter of transposition (although we must be careful to be consistent).
Note that it is possible to be both row *and* column stochastic.

2.1. Exercise. Give distinct examples of each of the following: row stochastic matrix; column stochastic matrix; row and column stochastic matrix.

[2]Brzezniak and Zastawniak, Basic Stochastic Processes, Springer 1999.

Suppose that the value of some variable (perhaps the price of a unit of pork bellies on the Chicago stock exchange) evolves randomly through a discrete series of time steps. If the probability of taking value x at time $n+1$ depends only on the value at time n, then this process is called a Markov chain. Suppose further that there are only finitely many values which the variable can take. Then we have a discrete time, finite state Markov process. Note that this process is determined by the set of transition probabilites between the various possible values. We arrange these into a matrix M, so that M_{ij} is the probability of taking value j at time $n+1$, given a value of i at time n. This means in particular that $\sum_j M_{ij} = 1$, since the probability of the variable taking a value, summed over all values, at time $n+1$ is 1.

Suppose we run this process for a long time. If it happens that the proportion of the time spent at each value settles down (irrespective of the initial value) then the vector ν of these proportions is called an invariant measure. If we arrange it as a row vector it will satisfy

$$\nu M = \nu$$

(in fact no stochastic matrix can have an eigenvalue with absolute value greater than 1; and they all obviously have 1 as an eigenvalue).

Our random walk on the web would simply choose to follow one of the links out of the present page at random. If we write δ for the diagonal matrix whose m-th diagonal entry is the number of links *out* of page m, then the appropriate stochastic matrix is

$$M = \delta^{-1}W.$$

It is possible that only a subset of all web pages can be reached following outgoing links from some particular starting point. To avoid getting trapped in this way we will also allow a $1 - p$ probability of simply ignoring the local links and jumping randomly to any point on the web. (We wouldn't want to do this totally random action any more than necessary, which is why we choose p fairly close to 1.) The stochatic matrix appropriate for this random jump mode on its own is simply the matrix f in which every entry is $1/\sigma$, where σ is the total number of pages on the web. Thus the combined matrix, allowing for the probability of chosing the random mode, is

$$M = (1-p)f + p\delta^{-1}W.$$

The totally random component makes sure that M does not break up into blocks which (via such beautiful ideas as the Peron–Frobenius theorem (see later)) ensures that the solution to the unit eigenvalue problem is unique (once the eigenvector is normalised so that the entries add to 1).

Let ι be the *vector* with all entries 1. Then $\nu^n \sigma f = \iota$ for ν^n any probability distribution row vector (exercise). It follows that, applying M to some probability distribution ν^n:

$$\nu^n M = \nu^n((1-p)f + p\delta^{-1}W) = \frac{(1-p)}{\sigma}\iota + p\nu^n\delta^{-1}W = \nu^{n+1}$$

confirming that M is the transition matrix for this process. The distribution is stable at ν:

$$\nu\left(1 - p\delta^{-1}W\right) = \frac{1-p}{\sigma}\iota$$

so

$$\nu = \frac{1-p}{\sigma}\iota + p\nu\delta^{-1}W$$

This means that the page rank vector is the invariant measure for the above random walk.

Since this is the eigenvector for the largest eigenvalue of the stochastic matrix we may compute it by starting with an (almost) arbitrary vector and simply repeatedly multiplying this by the stochastic matrix.

2.3 Examples

Model 1: Consider the very simple model of the web consisting of three pages, call them A,B and C, with A linking to B linking to C linking to A. This model has matrices

$$W = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \qquad \delta = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Then we have

$$M = (1-p)\left(\frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}\right) + p\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

This model has an obvious symmetry under permuting any two of the pages, so they must all be equally 'important'. This tells us that the invariant measure must be given by a vector with all entries equal. We can readily confirm this:

$$\left(\frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \end{pmatrix}\right) M = \frac{1-p}{3}\begin{pmatrix} 1 & 1 & 1 \end{pmatrix} + \frac{p}{3}\begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = \left(\frac{1}{3}\begin{pmatrix} 1 & 1 \end{pmatrix}\right.$$

(Note that we could have found the invariant measure by repeated multiplication on an arbitrary initial probability vector. In this case the steps in the iteration, although not the final answer, would have depended on p. Thus the rate of convergence to the final answer can depend on p. For an extreme example, considering the initial vector $\begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$ and $p = 1$ we see that this *never* converges!)

Model 2: Now consider what happens when B introduces a link back to A:

$$W = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \qquad \delta = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

so

$$M = (1-p)\left(\frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}\right) + p\begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

This model is connected so we can usefully consider $p \to 1$, whereupon

$$M = \begin{pmatrix} 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 1 & 0 & 0 \end{pmatrix}$$

By the argument above we may determine the (relative) PageRanks by repeatedly multiplying this matrix into an initial positive matrix. (To get the absolute ranks we need to normalise this initial matrix as a matrix of probabilities, but this normalisation is just a fixed overall factor so we will not worry about it for now.)

Consider

$$\begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 1 & 0 & 0 \end{pmatrix}^n = \begin{pmatrix} 1.5 & 1 & .5 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 1 & 0 & 0 \end{pmatrix}^{n-1}$$

$$= \begin{pmatrix} 1 & 1.5 & .5 \end{pmatrix} M^{n-2} = \begin{pmatrix} 1.25 & 1 & .75 \end{pmatrix} M^{n-3} = \begin{pmatrix} 1.25 & 1.25 & .5 \end{pmatrix}$$

$$= \begin{pmatrix} 1.125 & 1.25 & .625 \end{pmatrix} M^{n-5} = \begin{pmatrix} 1.25 & 1.125 & .625 \end{pmatrix} M^{n-6} = \dots$$

Can you see where this is going? Consider

$$\begin{pmatrix} x & x & y \end{pmatrix} M = \begin{pmatrix} x & x & y \end{pmatrix}$$

giving $x = \frac{1}{2}x + y$, $y = \frac{1}{2}x$, so putting $2x + y = 1$ (for probabilities) we have $x = \frac{2}{5}$, $y = \frac{1}{5}$.

Now explain this in terms of the linking and importance of the pages! At first site, A and B are clearly more important than C, but A gets votes from both B and C, so you might think it beats B. However, B votes for both A and C, so its vote for A is diluted, while A always votes entirely for B which, in this model, locks their importance levels together.

Exercise 2 *What happens if we make $p < 1$?*

Exercise 3 *Consider the model with four pages, A,B, C and D, and matrix*

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

Work out the PageRanks in case $p = .85$.

Exercise 4 *(Harder) Do a Google search for the common name "Paul Martin". Qualitatively explain the first page of results using PageRank.*

2.3.1 More exercises

Every simple loop free digraph is a representation of some model of the web.

Every $n \times n$ matrix with zeros down the diagonal and either 1 or 0 in each offdiagonal position is the adjacency matrix of some such digraph. Let A_n denote the set of such matrices. Evidently $|A_n| = 2^{n(n-1)}$. However, many of these matrices represent the same digraph up to isomorphism. (For example, any two such matrices containing precisely one nonzero element are isomorphic.)

A complete list of adjacency matrices of representative elements of isomorphism classes of 3 vertex simple loop free digraphs containing at most 2 edges is:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

We can obtain a 3 edge digraph by taking a 2 edge digraph and adding an edge. Starting with the first 2 edge digraph above we get

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$

but no others (up to isomorphism). Starting with the second 2 edge digraph we get one new digraph (up to isomorphism)

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Starting with the third 2 edge digraph we get one new digraph (up to isomorphism)

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

This is a complete list up to three edges.

We want to restrict attention to matrices with nonzero row sums. There are only two such in our list so far.

Exercise 5 *Compute the page rank vector (invariant measure) for each of these two cases. (Take $p = .85$ or another appropriate value of your choice.)*

Exercise 6 *Continuing with the three vertex case, list a complete set of representative elements with 4 or more edges, such that the matrix row sums are all nonzero. Compute page rank for every element of your list. (Take* $p = .85$ *or another appropriate value of your choice.)*

Exercise 7 *Give an example of a model of the web with 8 pages in which you can easily compute page rank (and compute it!).*

In order to study and compute with finite state Markov processes in general, and with our applications in particular, we see that we need familiarity with basic matrix theory. Here is a quick review.

2.4 Elementary Matrix Theory

Consider the eigenvalue equation for matrix M

$$Mv = \lambda v$$

The set of eigenvalues is the set of roots of the *characteristic equation*

$$P_M(\lambda) \ := \ |M - \lambda \mathbb{I}| = 0$$

Two matrices A, B are *similar* if they are related by a similarity transformation, i.e.

$$A = M^{-1} B M$$

for some M.

2.2. Exercise. Show that two similar matrices have the same characteristic equation, and hence the same eigenvalues.

In particular the trace and determinant of a matrix are not changed by similarity transformation.

Theorem 2.3. *[Cayley–Hamilton]*

$$P_M(M) = 0$$

It follows that M^n can be expressed as a linear combination of lower powers of M.

2.4.1 Norms

2.4. Definition. A norm on a vector space V is a real function $||v \in V||$ such that $||v|| \geq 0$; $||v|| = 0 \ \Rightarrow \ v = 0$; $||kv|| = k||v||$; $||v+w|| \leq ||v|| + ||w||$.

2.5. Example. The Euclidean norm is the $r=2$ case of $||v||_r = +\sqrt[r]{\sum_i |v_i|^r}$, and is a norm.

2.6. Definition. A matrix norm is a vector norm as above extended by the condition on conformable matrices M, N: $||MN|| \leq ||M||.||N||$.

Examples:

2.7. Definition. The Frobenius norm $||M||_F$ of a matrix M is defined by

$$||M||_F = +\sqrt{\sum_{i,j} |M_{ij}|^2} = +\sqrt{\text{Trace}((A^*)^t A)}$$

2.8. Exercise. (Optional) Show that the Frobenius norm is a matrix norm.

If $|| - ||$ is a vector norm on n-component column vectors then we get an 'induced' matrix norm on $n \times n$ matrices by

$$||M|| \ = \ \max_{||v||=1} \ ||Mv||$$

(NB, you need to convince yourself that the maximum exists!).

The matrix norm induced from $|| - ||_2$ is called the matrix 2–norm. Assume for a moment that M is real. Then to determine $||M||_2$ we need to determine the maximum value of

$$f(v) = ||Mv||_2^2 = v^t M^t M v$$

subject to $g(v) \ := \ v^t v = 1$. Form

$$h = f - \lambda g$$

(Lagrange multipliers). The system got by differentiating wrt each v_i is

$$(M^t M - \lambda \mathbb{I})v = 0$$

so λ must be such that $(M^t M - \lambda \mathbb{I})$ is singular (i.e. $P_{M^t M}(\lambda) = 0$). Thus

$$v^t (M^t M v) = \lambda v^t v = \lambda$$

and

$$||M||_2 = \overset{\max}{||v|| = 1} \ ||Mv|| = \overset{\max}{||v||^2 = 1} \ ||Mv||$$

$$= +\sqrt{\overset{\max}{v^t v = 1} \quad v^t M^t M v} = +\sqrt{\lambda_{max}}$$

where λ_{max} is the biggest such λ. Of course this is just the biggest eigenvalue of $M^t M$.

2.9. Definition. The spectral radius of a square matrix M is

$$\rho(M) = \max_{\lambda \in S(M)} |\lambda|$$

where $S(M)$ is the set of eigenvalues of M.

2.4.2 Jordan form

Let $S(M) = \{\lambda_1, .., \lambda_r\}$ be the set of eigenvalues of M, with multiplicities n_i. Let U_n be the $n \times n$ matrix with all entries zero except for those immediately above the main diagonal.

$$U_3 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Then $J_n(\lambda) = \lambda \mathbb{I}_n + U_n$ is called a Jordan block. Every $n \times n$ matrix is similar to a matrix of form

$$AMA^{-1} = \bigoplus_j \bigoplus_i J_{n_{i,j}}(\lambda_j)$$

where $\lambda_j \in S(M)$ and $\sum_i n_{i,j} = n_j$.

For the study of Markov processes we are interested in the behaviour of (certain) matrices when raised to a high power. Let us start by considering Jordan blocks.

2.10. Exercise. Show that if $J_n(\lambda)$ is a Jordan block with $n \geq 2$ then $(J_n(\lambda)^k)_{11} = \lambda^k$ and $(J_n(\lambda)^k)_{12} = k\lambda^{k-1}$. Determine the complete form of the k-th power.

If $\lambda < 1$ the limit $\lim_{k\to\infty}(J_n(\lambda))^k$ is the zero matrix. By the exercise above, the limit will not exist, indeed the matrix will diverge (certain entries grow bigger and bigger in magnitude as k increases) if $|\lambda| > 1$, or if $|\lambda| = 1$

and $n > 1$.

If $\lambda \neq 1$ while of unit magnitude and $n = 1$ then the diagonal term will oscillate, so that (although every power is finite) again there is no limit. The case $\lambda = 1$, $n = 1$ is obvious.

Note that for any transform A

$$AM^kA^{-1} = AMM^{k-1}A^{-1} = AMA^{-1}AM^{k-1}A^{-1} = (AMA^{-1})^k$$

Comparing these observations we see that $\lim_{k\to\infty} M^k$ will not exist unless $|\lambda_i| \leq 1$ for all i, and if $|\lambda_i| = 1$ for some k then $n_{ik} = 1$ for all i.

On the other hand, if M is stochastic then so is M^k for any k (in particular all entries remain in the probablistic interval). We deduce that a stochastic matrix has no eigenvalue with magnitude greater than 1. This is because, if M^k is finite then so is AM^kA^{-1} for any finite A, including the transform taking M (and hence also M^k) to Jordan form, so that if the Jordan form of M^k diverges (i.e. has entries of unboundedly large magnitude) we know M^k cannot be finite. By the same token we know that $n = 1$ in any Jordan block with $|\lambda| = 1$, so that each eigenvalue with magnitude 1 has an independent eigenvector associated to it.

2.5 The Perron–Frobenius Theorem

A matrix is *positive* if all its entries are positive.

The Perron–Frobenius theorem states that a finite positive matrix has a unique eigenvalue of largest magnitude, and this eigenvalue is positive; and that this eigenvalue has an eigenvector with all positive entries. [3]

A matrix M is *positivizable* if M^n is positive for some natural number n. It will be evident that the theorem applies to such matrices.

Let us prove the theorem in the special case of a matrix M in which the row sums are 1. The idea of this proof will be to consider $\overset{\min}{k} M_{kj}^n$ and $\overset{\max}{k} M_{kj}^n$ and show that these converge to the same thing as n gets large (which would force M_{ij}^n to settle down to a value independent of i). First we show that $\overset{\min}{k} M_{kj}^n$ (which we may take as zero for $n = 0$) increases with n:

$$M_{ij}^{n+1} = \sum_k M_{ik} M_{kj}^n \geq \overset{\min}{k} M_{kj}^n \sum_k M_{ik} = \overset{\min}{k} M_{kj}^n$$

[3] A proof of this theorem can be found, for example, in P P Martin, Potts models and related problems in statistical mechanics, World Scientific 1991.

Similarly

$$\max_k M_{kj}^{n+1} \leq \max_k M_{kj}^n$$

Now impose that M is positive. Let $e > 0$ be the smallest M_{ij}. Then

$$M_{ij}^{n+1} = \sum_k M_{ik} M_{kj}^n = \sum_k [M_{ik} - e M_{jk}^n] M_{kj}^n \quad + e \sum_k M_{jk}^n M_{kj}^n$$

$$= \sum_k [M_{ik} - e M_{jk}^n] M_{kj}^n \quad + e M_{jj}^{2n}$$

$$\geq \min_k M_{kj}^n \sum_k [M_{ik} - e M_{jk}^n] \quad + e M_{jj}^{2n} = (1 - e) \min_k M_{kj}^n \quad + e M_{jj}^{2n}$$

Since the RHS doesn't depend on i we have

$$\min_k M_{kj}^{n+1} \geq (1 - e) \min_k M_{kj}^n \quad + e M_{jj}^{2n}$$

Similarly

$$\max_k M_{kj}^{n+1} \leq (1 - e) \max_k M_{kj}^n \quad + e M_{jj}^{2n}$$

and hence

$$\max_k M_{kj}^{n+1} - \min_k M_{kj}^{n+1} \leq (1 - e)(\max_k M_{kj}^n - \min_k M_{kj}^n)$$

so the sequence of maxima and the sequence of minima have the same limit. Let us call it π_j. For any i

$$\min_k M_{kj}^n \leq M_{ij}^n \leq \max_k M_{kj}^n$$

so

$$M_{ij}^{\infty} = \pi_j > 0$$

independent of i.

We have a matrix of form

$$M^{\infty} = \begin{pmatrix} \pi_1 & \pi_2 & \pi_3 \\ \pi_1 & \pi_2 & \pi_3 \\ \pi_1 & \pi_2 & \pi_3 \end{pmatrix}$$

and

$$M^{\infty} M = M^{\infty}$$

so in particular

$$\begin{pmatrix} \pi_1 & \pi_2 & \pi_3 \end{pmatrix} M = \begin{pmatrix} \pi_1 & \pi_2 & \pi_3 \end{pmatrix}$$

giving us our invariant measure.

The trace of M^∞ is $\sum_i \pi_i = 1$. This is the sum of all eigenvalues of magnitude 1, but if there were any such eigenvalues not equal to 1 the limit would not exist. Indeed the rank of M^∞ is 1, so all it's eigenvalues except one have value zero.

2.11. Exercise. Find the eigenvalues and eigenvectors for each of the following:

$$\frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \qquad \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \\ 2 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & \\ & & & 1 \\ 1 & & & \end{pmatrix} \qquad \left(\begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ & & \\ 0 & 1 & \end{matrix}\right.$$

Chapter 3

Coding Theory

Preamble

The aim of this chapter is to study coding theory. We begin with a few general words about what coding theory *is*, and why we want to study it (i.e. what is it good for?). [1]

Coding is the act of preparing information for transmission.[2]

[1] Warning: This preamble is mildly philosophical in nature. It might be best to skip it for now, and come back after you have studied a few chapters of coding practice.

[2] For example, from Dictionary.com:
`http://dictionary.reference.com/browse/coding` we have:
11. Linguistics. a. the system of rules shared by the participants

There are many subtleties to this definition. For example, in order to verify that information exists it has to be transmitted, so coding is effectively part of the creation of information. Anyway, from the pseudo-definition above it will already be clear that coding is 'important'. It also contains language as a substructure, which further emphasises its importance.[3]

All transmission carries the risk of corruption. The *Science* (or Theory) of coding is concerned with minimising this risk (in some, usually quantitatively probabilistic, sense).

Example: suppose we need to be certain a message has got through exactly as sent (e.g. a 'zipped tar' file). [4] What can we do?

in an act of communication, making possible the transmission and interpretation of messages.

[3]See for example
`http://leoalmanac.org/journal/Vol_14/lea_v14_n05-06/lglazier.asp`

[4]What does 'certain' mean here?! This is another Statistics-meets-Physics/Philosophy question...

Methodology, caveats and acknowledgements

As you can see, intrinsic to this chapter are notions of communication, data, risk, and information. These are not trivial notions, and we won't be able to define them up-front. In mathematics we try to keep the number of terms that are used without definition to a minimum. This is because every term used without definition is a possible source of confusion between person A (the propagator, perhaps, of an idea) and person B (the recipient). Person A simply has to trust that person B is understanding the same thing by her term. If not, then any idea built on it will be flawed. Unfortunately it is never possible to define *all* terms. In mathematics, for example, we generally take on trust that others understand the same thing by the term 'set' as we do.

In the *applications* of mathematics, however, this 'define everything' discipline can conflict with progress. Our strategy will be to use some terms, where necessary, without an initial definition; but to try to come back to them later and find a way to check that we really do agree on their meaning.

To begin with, then, we may consider *communication* as the process whereby some ‘data’ held in one ‘machine’ is passed so that a copy is held in some other machine. This is somewhat analogous to the process whereby an idea held in your mind might be communicated to me, so that I then hold that same idea. [5] The extent to which this analogy works (or, if failing, we still want to treat both processes) is a matter for discussion. It is probably true to say that we can work more comfortably with the first process than the second, but the second is ultimately perhaps more interesting?

[5]Descartes doubts even that other people exist, so communicating with them is something not to be taken lightly, if we are being really careful! We simply can’t afford to be this careful here — we have concrete applications to address.

I thank Martin Speight for lending me his own beautiful notes on Coding Theory, which have been invaluable in the preparation of this Chapter.

Some recommended reading:

R Hill, ”First Course in Coding Theory”, Clarendon Press, 1986.

G A Jones and J M Jones, “Information and Coding Theory”, Springer, 2000.

Figure 3.1: A coding theory class.

Coding theory in a single picture

First you need to look at figure 3.1. But then... See figure 3.3. Here person **A** tries to communicate the result of a football match (Win, Lose or Draw). This is done by: (1) setting up an 'encoding' of the set of possible messages (W,L,D) — in this case by associating them with different points on the whiteboard; (2) transmitting the match result down a noisy channel — in this case by pointing at it. (This communication method might

Figure 3.2: A noisy channel.

not be a very *good* practical communication method under the circumstances, but it contains nice analogies of many of the key points of coding theory.) All these ideas will be explained as we go along.

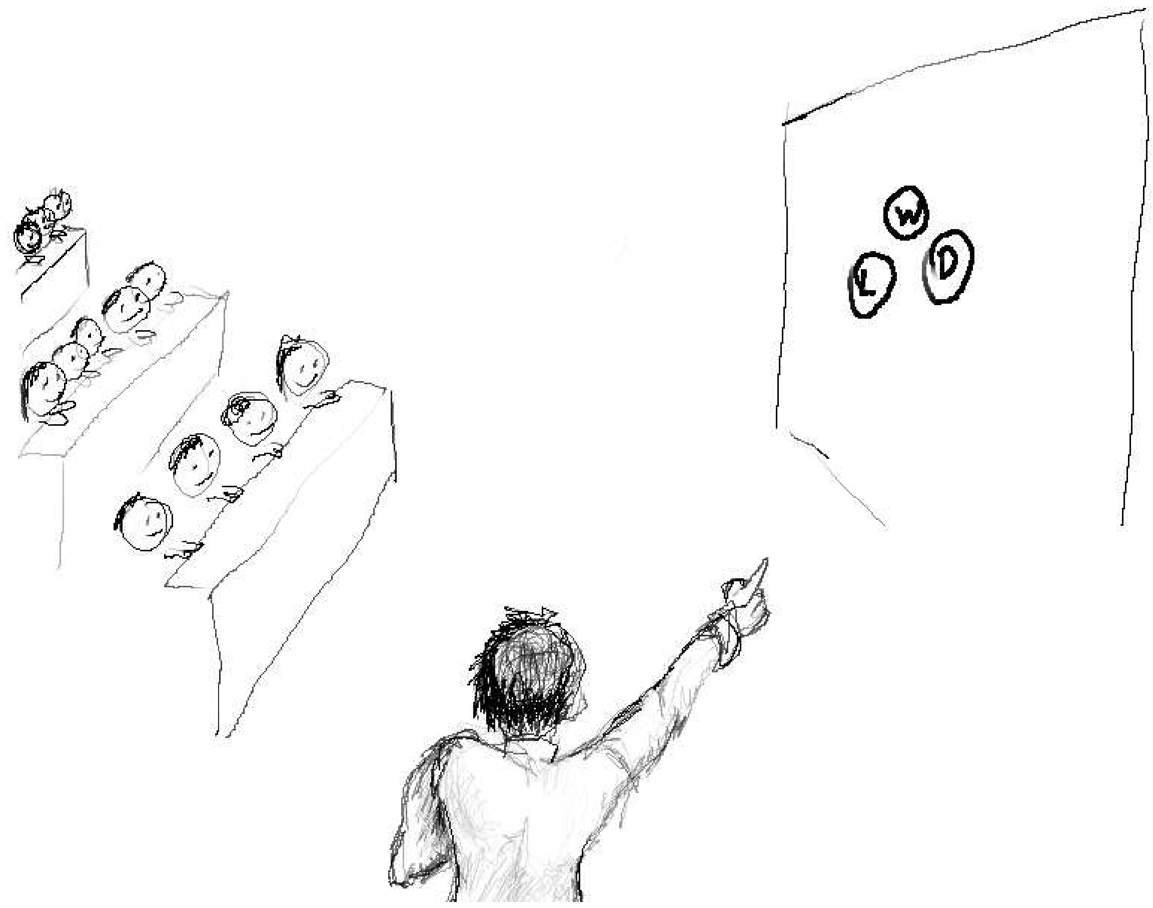


Figure 3.3: Transmitting data through a noisy channel.

3.1 Coding

A coding is a representation of data. (What is data?...)

Let S be a set. A sequence of elements of S of length l is an element of

$$S^l = S \times S \times ... \times S$$

For example, if $S = S_{alph}$ is the usual 26 element alphabet then

$$(w, i, l, l, y, o, u, m, a, r, r, y, m, e)$$

is a sequence of length 14. (Where no ambiguity arises we might drop the brackets and commas.)
A *finite* sequence is a sequence of finite length.

Define S^0 to be the empty sequence, and

$$S^* = \cup_{l \geq 0} S^l \qquad \text{and} \qquad S^+ = \cup_{l > 0} S^l$$

Define a product on S^* by

$$\begin{aligned} \circ S^* \times S^* &\rightarrow S^* \\ (x, y) &\mapsto x \circ y = xy \end{aligned}$$

where xy is the concatenation of x and y.

3.1. Example. If $x = 01010110$ and $y = 1$ then $xy = x1 = 010101101$.

A significant percentage of all human wisdom (?!), and all human communication, has been encoded as sequences using a mild generalisation of the alphabet S_{alph}. [6]

'Data', for our present purposes, takes the form of some finite sequence. We assume that this sequence has value to us for some reason (determining the humanistic value of a given sequence is beyond the scope of this section, but it might contain, for example, a list of transactions in your bank account for the last year). The challenge we face is to transmit this data to a new location.

[6]On the other hand there is no system which will enable us to encode even a single 'generic' element of the set $(0, 1)$ (the unit open real interval).

Some elements in this interval can be communicated by more abstract means. For example π, e and $\sqrt{2}$. Such abstractions are presently among the features distinguishing humans from computers... but that is another story.

For example, perhaps a person in England wants to communicate the question implied by the sequence $(w, i, l, l, y, o, u, m, a, r, r, y, m, e)$ to a friend in Australia. In this case obviously shouting it out, or writing it onto a sheet of paper and throwing this in a southerly direction, is not going to get the job done, even if the recipient knows to expect a message (audible or written, respectively) in some given time-window. Phoning or sending an email might work better. But all these efforts can be considered as involving the same basic process:

1. Source (person A) has a message to communicate (I want to offer marriage to person B). We shall assume that the source has this message expressed as a finite sequence in some source alphabet T.

2. Source message is encoded in some way suitable for travel to B (for example by vocalising in spoken English — whatever that is). We shall assume that the encoding passes the message to a sequence in a not necessarily distinct coding alphabet S.

3. Encoded version travels somehow to B, degrading gradually for various reasons as it travels;

4. Degraded encoded version reaches target's decoder (nominally in our example it is a sound, so the decoder is an ear/brain system; but obviously the sound heard by B at the appropriate point in time will have only a negligible amount of correlation with the original encoding). An attempt is made to decode this version.

5. Some approximation to the original message arrives for use at the target.

3.2. A code C for a source alphabet T is a function $f : T \to S^+$ to sequences in code alphabet S. The properties of codes that we shall focus on depend on the image set $f(T)$ rather than the details of the map itself, so one often regards a code simply as this set of words.

The *extension* of C to T^* is obtained simply by using f to encode each symbol in succession.

3.3. Example. (i) If $f(a) = 001$ and $f(b) = 010$ then $f(abba) = 001010010001$.

(ii) If $f(a) = 1$ and $f(b) = 010$ then $f(abba) = 10100101$.

We shall be interested in fixed-length codes (a discussion of variable length codes can be found for example in Jones and Jones (Springer SUMS, 2000)):

3.4. Definition. A block code

$$C = \{(x_1, x_2, ..., x_n), (y_1, y_2, ..., y_n), ...\}$$

of length n over set S is a subset of S^n. Code C is q-ary if $|S| = q$.

An encoding is a recasting of one code as another (or the encoding of a message, but no usable message is really entirely unencoded).

3.5. Example. Let S, T be sets and

$$f : T \to S^l$$

Then a code $C \in T^n$ can be coded over S by applying f to each element of each sequence x in turn as before. This time:

$$f : T^n \to S^{nl}$$

where $f(x)_{in+j} = f(x_i)_j$ for $j < |S|$.

In particular (1) if $T = \{N, S, E, W\}$ and $S = \{0, 1\}$ and $f_1(N) = (0, 0) = 00$, ..., $f_1(W) = (1, 1) = 11$ then

$$f_1(EESW) = f((E, E, S, W)) = 10100111$$

(2) if T, S as above and $f_2(N) = 000$, $f_2(S) = 011$, $f_2(E) = 101$, $f_2(W) = 110$, then

$$f_2(EESW) = 101101011110$$

3.2 Transmission

Now suppose we transmit the message $EESW$ — in any invertible encoding.
We assume that the recipient knows (1) that the original message was some sequence in $\{N, S, E, W\}$, and (2) how we encoded it (if at all).
Thus, if the encoded message arrives intact, she can invert the encoding to recover the original message.
BUT We want to consider the realistic scenario in which, with some probability, part of the encoded message is corrupted in transmission.
We want to ask: What can be done about that? And what can 'best' be done?

For example, suppose that there is a 1% chance that recipient B will mishear any term in the sequence in the original encoding. Then there is a roughly 4% chance that the message will arrive with a corrupted element.

Note that there is no way for the recipient to tell whether the message has been corrupted or not, in the original encoding or in f_1 (from Example 3.5).

In f_2, however, not every binary code of length 3 is the image of an element of T, so if 101 was corrupted to 001, say, we would know at least that there had been an error in transmission. Indeed with this encoding *every* single element transmission error would show up. However double errors could still appear to be OK.

Now consider

(3): T, S as in Example 3.5 above and $f_3(N) = 00000$, $f_3(S) = 01101$, $f_3(E) = 10110$, $f_3(W) = 11011$.

3.6. Exercise. Verify that if any two errors occur then the received message is not the image of any sent message, signaling an error.

Further, if a single error occurs the sent message is recoverable anyway. For example suppose $E \mapsto 10110 \to 10010$ after transmission. We cannot decode this, but considering the following table of number of places differing from the encoding of each element of T:

encoding	places differing
00000	2
01101	5
10110	1
11011	2

we guess correctly that the intended element was E.

We say that (3) is 2 error *detecting*; or single error *correcting*.

Note that the cost of these improvements was higher block length, which introduced some redundancy. That is, we have a trade-off between efficiency and reliability.

3.3 Hamming distance

Let us try to be more precise about this error-correcting facility. Recall

3.7. Definition. Let S be a set. A map $d : S \times S \to \mathbb{R}$ is a metric on S if it satisfies: (i) $d(x,y) = 0 \iff x = y$
(ii) $d(x,y) = d(y,x) \forall x, y \in S$
(iii) $d(x,y) \leq d(x,z) + d(z,y) \forall x, y, z \in S$ (triangle inequality).

Note that the usual distance in Euclidean space $\mathbb{R}^n$ is a metric. We don't have numbers (necessarily) in our 'alphabets', so our basic distance function is cruder:

3.8. Definition. Given $x, y \in S^n$ the (Hamming) distance between them is $d(x,y) =$ number of positions in which x, y differ.

3.9. Proposition. *The Hamming distance is a metric.*

Prove it!

3.10. Definition. The minimum distance of a code $C \subset S^n$ is

$$d(C) = \min\{d(x,y) | x, y \in C, x \neq y\}$$

Examples:

C_1	00	01	10	11
00		1	1	2
01			2	1
10				1
11				

so that $d(C) = 1$ in case (1). Similarly in case (2) above the min distance is 2; and in case (3) it is 3. (Exercises!)

3.11. Proposition. *(a) If $d(C) \geq t+1$ then C can detect up to t errors;*
(b) If $d(C) \geq 2t+1$ then C can correct up to t errors by the 'pick the closest' strategy.
Proof: Exercise, or see below.

3.12. Definition. For any $x \in S^n$ and $r \in \mathbb{N}$ the ball of radius r (or r-ball) centred on x is

$$B_r(x) := \{y \in S^n | d(x,y) \leq r\}$$

That is, the set of sequences that differ from x in no more than r places.
An r-sphere is

$$S_r(x) := \{y \in S^n | d(x,y) = r\}$$

That is, the 'outer shell' of an r-ball.

3.13. Let $C \subset S^n$. Consider the collection of t-balls centred on all $x \in C$. This is a 'fuzzy picture' of the elements x. Each is surrounded by the area of uncertainty in it, in a neighbourhood of S, caused by up to t transmission errors.

This gives us a kind of picture for the proof of (3.11) (see Figure 3.4):

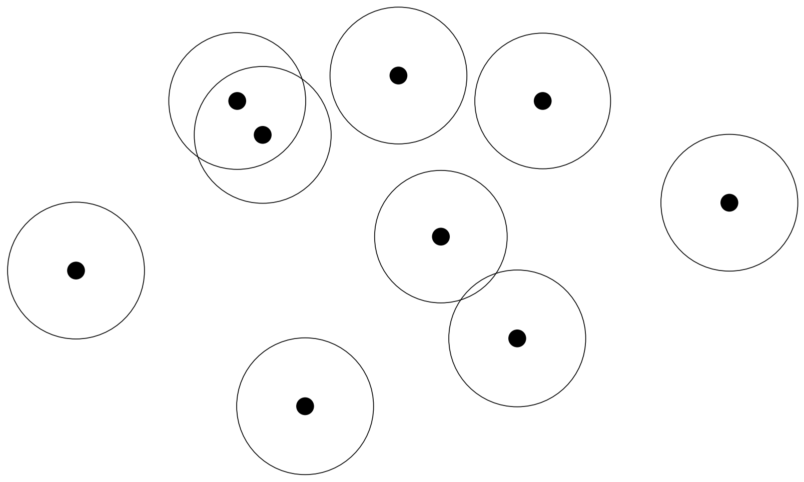


Figure 3.4: Ball packing heuristic (using Euclidean metric).

(a) If $d(C) \geq t+1$ then no x lies in another's ball. Thus if 1 up to t errors occur then the received message is not in C and we *know* we have an error.
(b) If $d(C) \geq 2t+1$ then even the balls are disjoint (this is perhaps not so obvious with the Hamming distance, cf. say the usual Euclidean metric, but the triangle inequality is what we need to confirm it), and if 1 up to t errors occur then the received message is closer to x than any other $y \in C$. $\square$

3.4 Optimisation

3.4.1 Code choice affects transmission error probability

We are making a case, superficially, that f_3 is more reliable than f_1 when transmitting over a channel with errors. But in replacing by sequences 2.5 times as long we are giving it far more digits to get wrong! Is f_3 really more reliable? Less reliable? Does it really make any difference?

To settle this we need to compute a probability for a message being wrongly decoded in each case.

In order to do this it is simplest to make some assumptions about error probabilities in the transmission 'channel':

(a) Each transmitted digit is equally likely to be corrupted, with probability p.

(b) If a digit is corrupted, any of the $q-1$ other letters in S are equally likely to occur.

This is a q-ary symmetric channel, with symbol error probability p.

Sending symbol S in the f_1 code we send 01. It will be decoded correctly *only* if no errors occur:

$$P_{corr}(01) = (1-p)^2$$

so the error probability is

$$P_{err}(01) = 1 - (1-p)^2 \tag{3.1}$$

In f_3 we send 01101. This will decode correctly if 0 or 1 errors occur (possibly more) so

$$P_{corr}(01101) \geq (1-p)^5 + 5p(1-p)^4$$

so

$$P_{err}(01101) = 1 - (1-p)^5 - 5p(1-p)^4 \tag{3.2}$$

If p is small then (3.2) is much smaller than (3.1). E.g. if $p = 0.01$ then $P_{err}(01) = .0199$ while $P_{err}(01101) \leq .0009801496$. So increasing word length by 2.5 times reduced error probability 20-fold!

If p is bigger then f_1 doesn't look so bad (for example at around $p = .4$ and above it is better than f_3).

Anyway, the point is it makes a difference. So the science of coding theory is non-trivial. The game is ON!

3.4.2 All possible block codes

3.14. Definition. A q-ary (n, M, d)-code is a block length n code with M codewords and minimum distance d.

For S a set let $P(S)$ denote the power set of S. Thus $P(S^n)$ is the set of length-n $|S|$-ary codes; and a q-ary (n, M, d)-code C is an element of $P(S^n)$ (some S of degree q) such that $|C| = M$ and $d(C) = d$.

As a convention, by default we assume that if $|S| = q$ then

$$S = \{0, 1, ..., q-1\}$$

Write (n, M, d)-cod$_q$ for the set of q-ary (n, M, d)-codes (or just (n, M, d)-cod if q is fixed). Thus:

$$P(S^n) = \sqcup_{M,d}(n, M, d)\text{-cod}$$

3.4.3 Achievable code parameters

Define

$$A_q(n,d) = \max M, \text{ for fixed } q,n,d$$

that is, the size of the largest possible q-ary (n, M, d)-code. Since q, n determine the size of the ‘space’ in the picture we considered earlier, and d the size of the ‘exclusion zone’ around each point — a ball in that space, it is reasonable that only so many such balls can be fitted in the space without overlap.

The following gives an upper bound on $A_q(n,d)$:

Theorem 3.15. *(Singelton bound) For any q-ary (n,M,d)-code, $M \leq q^{n-(d-1)}$. Hence $A_q(n,d) \leq q^{n-(d-1)}$.*

Proof: Let C be such a code, with code alphabet S, and $\pi : C \to S^{n-(d-1)}$ be the map

$$\pi : (x_1, x_2, .., x_n) = (x_1, x_2, .., x_{n-(d-1)})$$

Take $x \neq y \in C$. If $\pi(x) = \pi(y)$ then x, y agree in $n-(d-1)$ places and hence differ in at most $d-1$. But then $d(x,y) \leq d-1$. Hence π is one-to-one. Hence its domain is no larger than its codomain:

$$M = |C| \leq |S^{n-(d-1)}| \; = \; q^{n-(d-1)}$$

□

This is not usually a very good bound, but is saturated in some circumstances.

3.16. Example. What is $A_2(3,2)$? By the singleton bound

$$A_2(3,2) \leq 2^{3-(2-1)} = 2^2 = 4$$

But our example (2) is a 2-ary (3,4,2)-code, so $A_2(3,2) \geq 4$. Hence $A_2(3,2) = 4$.

A much better upper bound is generally given by the 'ball packing argument'. This is built on a consideration of the amount of 'space' occupied by the 'error ball' around a codeword transmitted with a given number of errors:

3.17. Lemma. *If $x \in S^n$ then*

$$|B_t(x)| = \sum_{r=0}^{t} \binom{n}{r} (q-1)^r$$

Proof: $|S_r(x)|$ is the number of strings in S^n differing from x in precisely r places. This is product of the number of ways to pick the r differing places with the number of ways to assign a differing digit in each place:

$$|S_r(x)| = \binom{n}{r} (q-1)^r$$

□

Theorem 3.18. *(Ball packing bound) Let C be a q-ary (n, M, d)-code with $d \geq 2t + 1$. Then*

$$M \sum_{r=0}^{t} \binom{n}{r} (q-1)^r \leq q^n$$

Proof: Since $d \geq 2t+1$, the t-balls centred on codewords are all disjoint. Hence

$$| \cup_{x \in C} B_t(x)| = \sum_{x \in C} |B_t(x)| = M \sum_{r=0}^{t} \binom{n}{r} (q-1)^r$$

by Lemma 3.17. But

$$(\cup_{x \in C} B_t(x)) \subset S^n \ \Rightarrow \ | \cup_{x \in C} B_t(x)| \leq |S^n| = q^n$$

□

We can use this bound to rule out the existence of codes with certain properties. For example, there is no 3-ary (6,10,5)-code, since, with $t = 2$ ($d = 2 \times 2 + 1$)

$$M \sum_{r=0}^{t} \binom{n}{r} (q-1)^r = 730$$

while $q^n = 3^6 = 729$.

However, even if $q, (n, M, d)$ passes the BP bound it does not *follow* that a code exists. For example, there is no 2-ary (6,9,4)-code, even though

$$M \sum_{r=0}^{t} \binom{n}{r} (q-1)^r = 9(1+6) = 63 < 64 = q^n$$

In this case we can actually rule out a code using the singleton bound:

$$q^{n-(d-1)} = 2^{6-3} = 8$$

while $M = 9$. But even if $q, (n, M, d)$ passes both bounds it does not follow that such a code exists. (See table 3.1 for example.)

n	$d=3$			$d=5$		
	actual	*singleton*	*ball−packing*	*actual*	*singleton*	*bp*
5	4	8	5	$2*$	2	$2*$
6	8	16	9	$2*$	4	$2*$
7	$16*$	32	$16*$	2	8	4
8	20	64	28	4	16	6
9	40	128	51	6	32	11
10	$72-79$	256	93	12	64	18
11	144	512				
12	256	1024				
13	512	2048				
14	1024	4096	1092	128	1024	154
15	$2048*$	8192	$2048*$	256	2048	270
16	$2560-3276$	16384	3855	$256-340$	4096	478
17	$\geq 83*2^6$					
$\vdots$						
47	$\geq 9*2^{48}$					
$\vdots$						
163	$\geq 19*2^{151}$					

Table 3.1: Table of known values for $A_2(n,d)$, and some bounds. (See R Hill, A First Course in Coding Theory; or N Sloane's online page: http://www.research.att.com/∼njas/codes/And/. The

Which value of t do we use in the BP bound? The largest t such that $2t+1 \leq d$, that is, $t \leq (d-1)/2$.

The largest integer not exceeding $z \in \mathbb{R}$ is written $\lfloor z \rfloor$ ('Floor function'). So use

$$t = \lfloor \frac{1}{2}(d-1) \rfloor$$

So the BP theorem implies

$$A_q(n,d) \leq \lfloor \frac{q^n}{\sum_{r=0}^{\lfloor (d-1)/2 \rfloor} \binom{n}{r}(q-1)^r} \rfloor$$

since $A_q(n,d)$ is an integer by definition. (These are the values tabulated under ball-packing.)

Note that the collection of t-balls is disjoint. If they completely cover S^n this is obviously the best use of the 'space' we can make, and the code is said to be *perfect*.

3.19. Definition. A q-ary (n, M, d)-code is perfect if the collection of t-balls centred on codewords, $t = \lfloor (d-1)/2 \rfloor$, is a partition of S^n.

Note that this happens if and only if equality occurs in Theorem 3.18.

Note also that this cannot happen if d is even (exercise).

3.20. Example. For our existing examples:

(1) is trivially perfect.

(2) $d = 2$ is even, so not perfect.

(3) is a 2-ary (5,4,3)-code:

$$M \sum_r \binom{n}{r} (q-1)^r = 4(1 + \binom{5}{1} 1) = 24$$

while $|S^5| = 2^5 = 32$, so not perfect.

3.21. Another kind of error-robust code, favoured by deaf people such as the author (!)[7], is a repetition code. A binary repetition code of length $n = 2t + 1$ is

$$C = \{00...0, 11...1\}$$

Clearly this is a $(2t + 1, 2, 2t + 1)$-code.
Every string $y \in S^{2t+1}$ either has more 0s than 1s, implying $y \in B_t(00...0)$;
or more 1s than 0s, implying $y \in B_t(11...1)$.
Hence $S^{2t+1} = B_t(00...0) \sqcup B_t(11...1)$.

3.22. Now, why did we only include the d odd cases in our table of $A_2(n, d)$?

For $A_2(n, d)$ we can deduce the even d cases from the odd.

[7] and roadies

3.23. Definition. The weight of a string $x \in S^n$ is

$$w(x) = \#\text{non-zero entries in } x$$

E.g. $w(011) = 2 = w(10010)$.

3.24. Lemma. *Suppose $S = \{0, 1\}$ and $x, y \in S^n$ both have even weight. Then $d(x, y)$ is even.*

Proof: Let $\underline{n} = \{1, 2, ..., n\}$ and, fixing x, y,

$$\underline{n}_{ij} = \underline{n}_{ij}(x, y) = \{k \in \underline{n} \mid x_k = i \text{ and } y_k = j\}$$

For example if $x = 01101$, $y = 10110$ then $\underline{n}_{00} = \emptyset$ and $\underline{n}_{01} = \{1, 4\}$.

(We will give the proof in the binary case as stated. Generalisations of the result are possible. Formulation of a suitable statement is left as an exercise (but will not be needed here).)

Now $w(x) = |\underline{n}_{10}| + |\underline{n}_{11}| = 2l$ for some l, since $w(x)$ is even; and $w(y) = |\underline{n}_{01}| + |\underline{n}_{11}| = 2m$ for some m similarly. Thus

$$d(x, y) = |\underline{n}_{10}| + |\underline{n}_{01}| = 2l + 2m - 2|\underline{n}_{11}|$$

□

3.25. Definition. An q-ary (n, M, d)-code is optimal if $M = A_q(n, d)$.

For $k \in \{1, 2, ..., n\}$ define 'projection'

$$\pi_k : S^n \to S^{n-1}$$

by $x \mapsto \pi_k(x) = x_1 x_2 ... x_{k-1} x_{k+1} ... x_n$ (deleting the k-th digit). This also acts, by restriction, on any subset of S^n, and hence on any code $C \in P(S^n)$, to produce a new code $\pi_k(C) \in P(S^{n-1})$.

For $i \in S$ define 'projection onto $x_k = i$-hyperplane' (abusing notation as if S^n were $\mathbb{R}^n$)

$$\pi_k^i : S^n \to S^n$$

by $x \mapsto \pi_k(x) = x_1 x_2 ... x_{k-1} i x_{k+1} ... x_n$ (replacing the k-th digit by i).

Note that if $D \in (n, M, d)$-cod with $d > 1$ then $|\pi_k(D)| = M$, since the maximum reduction in distance between distinct points caused by deleting one letter is 1 (so distinct points are still distinct after projection). That is

$$\pi_k : (n, M, d+1)\text{-cod} \to \sqcup_{d' \in \{d, d+1\}} (n-1, M, d')\text{-cod}$$

Theorem 3.26. *Suppose d odd. A 2-ary (n, M, d)-code exists iff a 2-ary $(n+1, M, d+1)$-code exists.*

Proof: (i) (Only if part): Let $C \in (n, M, d)$-cod. We construct $C' \in (n+1, M, d')$-cod (some d') as follows.

For each $x \in C$ let $x' = x0$ if $w(x)$ even and $x' = x1$ if $w(x)$ odd.

Note that $d \leq d' \leq d+1$. But every x' has even weight by construction so d' is even by Lemma 3.24. Hence $d' = d+1$.

(ii) (If part): Let $D \in (n+1, M, d+1)$-cod$_2$. Take $x, y \in D$ such that $d(x, y) = d+1$. Find a digit, the k-th say, where they differ. Construct $D' \in (n, M, d')$-cod$_2$ by $D' = \pi_k(D)$. Note that $d \leq d' \leq d+1$. But $d(x', y') = d(x, y) - 1 = d$. Hence $D' \in (n, M, d)$-cod$_2$. $\square$

Corollary: If d odd then $A_2(n+1, d+1) = A_2(n, d)$.

3.27. Lemma. $A_q(n, d+1) \leq A_q(n, d)$.

Proof: Let C be an optimal $(n, M, d+1)$-code, so $M = A_q(n, d+1)$. Choose $x, y \in C$ with $d(x, y) = d+1$. Assume x, y differ in k-th digit. Remove x from C and replace it with x':

$$x' = \pi_k^{y_k}(x)$$

New code C' contains x' and y and $d(x', y) = d$ by construction, so $d(C') \leq d$. Let $z, w \in C'$. If neither is x' then $z, w \in C$ so $d(z, w) \geq d+1 > d$. If $z = x'$ (say) then

$$d+1 \leq d(x, w) \leq d(x, x') + d(x', w) = 1 + d(z, w)$$

so $d(z, w) \geq d$. Thus $d(C') \geq d$, so $C' \in (n, M, d)$-cod, so $A_q(n, d) \geq M$. □

This gives us one last bound on $A_q(n, d)$:

Theorem 3.28. $A_q(n+1,d) \leq qA_q(n,d)$

Proof: Let C be an optimal q-ary $(n+1, M, d)$-code. Define $C_i = C \cap \pi^i_{n+1}(C)$. Clearly $C = \sqcup_{i \in S} C_i$ so $M = |C| = \sum_{i \in S} |C_i|$. Thus at least one of the C_is has order at least M/q. Choose such a C_i ($i = k$, say) and construct C' from it by deleting the last digit of each codeword:

$$C' = \pi_{n+1}(C_k)$$

Since $C_k \subset C$ we have $d(C_k) \geq d(C) = d$. But $d(C') = d(C_k)$ since all codewords in C_k agree in the last digit. Hence C' is a q-ary (n, M', d')-code with $M' \geq M/q$ and $d' \geq d$, so $A_q(n, d') \geq M/q$. But $d' \geq d$ so by (iterated use of) the Lemma above

$$A_q(n,d) \geq A_q(n,d') \geq M/q = A_q(n+1,d)/q$$

□

3.29. Example. Given $A_2(10,3) \leq 79$ it follows that $A_2(11,3) \leq 2 \times 79 = 158$.

3.30. Exercise. Use the above theorem to give an alternative proof of the singleton bound.

3.4.4 More exercises

3.31. Exercise. For each of the following triples (n, M, d) construct, if possible, a binary (n, M, d) code:

$$(6, 2, 6) \qquad (3, 8, 1) \qquad (4, 8, 2) \qquad (8, 40, 3)$$

If no such code exists, prove it.
Answer:
A q-ary repetition code has $M = q$ and $d = n$ for any q, n. Our first case is an example of this: $\{000000, 111111\}$ is a (6,2,6) code.
As we have set things up, all codewords are necessarily distinct. This means that d is necessarily at least 1. To make a $d = 1$ code, then, all we have to do is make any code at all. The biggest q-ary length n code has $M = q^n$ (just include every possible codeword). For binary $n = 3$, therefore, this biggest code has $M = 8$. That is, for (3,8,1):

$$\{000, 001, 010, 011, 100, 101, 110, 111\}$$

is the unique such code.
For our third case we can use the parity idea (proof of Theorem 3.26) to increase the distance by 1 from our (3,8,1) code:

$$\{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}$$

For our fourth case it is no longer obvious how to construct a code. Under the circumstances it is prudent to check if such a code is impossible, by checking the BP and singleton bounds. In this case one finds that the BP bound fails, so there is no such code.

An (undirected) graph G is a set V_G of vertices together with a set E_G of edges between them (for a more careful definition see Chapter 5). A *complete* graph is a graph in which every pair of vertices is connected by one edge.

A graph morphism $\phi : G \to G'$ is a map $\phi : V_G \to V_{G'}$ such that $(v_1, v_2) \in E_G$ implies $(\phi(v_1), \phi(v_2)) \in E_{G'}$.

3.32. Exercise. Consider the graph $G(n,k)$ each of whose vertices is a 2-ary sequence of length n; with an edge (x,y) whenever $d(x,y) \geq k$. A 2-ary length n code C is any subset of the vertex set of $G(n,k)$. If $G(n,k)$ restricts to the complete graph on C then $d(C) \geq k$.

(a) Prove it!

(b) Write down a maximal complete subgraph of each of the following: $G(3,3)$, $G(4,3)$, $G(5,3)$.

(c) If there is a complete graph of order l in $G(n,k)$ (l vertices) then there is a complete graph of order l including the vertex 000...0. Prove it.

(d) Let $\Psi : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$ denote swapping the first two entries in the sequence (e.g. $\Psi(10111) = 01111$). Then Ψ defines a graph homomorphism from $G(n,k)$ to itself. Prove it. (Can we say more?)

ANSWERS: (a) Try this yourself. Note that it says that $A_2(n,k)$ is the size of a maximal complete subgraph in $G(n,k)$.
(b) We give our complete graph as a list of vertices in each case:
$G(3,3)$: $\{000, 111\}$ (equally good would be $\{001, 110\}$, but it will be clear than neither subgraph can be enlarged without losing the completeness property);
$G(4,3)$: $\{0000, 1110\}$;
$G(5,3)$: $\{00000, 11100, 10011, 01111\}$.
(c) If we change the first entry in every vertex sequence in $G(n,k)$ (from 0 to 1 or from 1 to 0) then the Hamming distances between vertices are not changed. The same applies if we change any given entry in every sequence simultaneously. In this way we may take any vertex (in a complete subgraph, say) and change it to 000...0 without changing the edges in the subgraph (so it remains as the complete graph). $\square$
(d) For every pair of vertices $d(x,y) = d(\Psi x, \Psi y)$, since the first two entries are interchanged in *both*. In fact Ψ gives a graph isomorphism of $G(n,k)$ with itself. But of course Ψ would not fix some arbitrary subset C in general.

3.5 Finite fields

We have repeatedly thought of S^n as if it were something like $\mathbb{R}^n$, that is, as if it were a vector space, and $C \subset S^n$ a vector subspace. Now we want to go further and think of strings

$$x = x_1 x_2 ... x_n = (x_1, x_2, ..., x_n)$$

as vectors, so that we can add them, and multiply by scalars.

In its simplest form this means that we want S itself to be like $\mathbb{R}$, in the sense of having addition and multiplication defined (perhaps even subtraction, and division by 'non-zero' elements). But S cannot *be* $\mathbb{R}$, since it is finite.

The composition requirements are summarised by saying that we want S to be a field. We should recall the definition of field; and then see if we can think of any *finite* fields that we could use for our alphabet.

The definition of field is quite long. We can break it up a little into stages:

3.33. Definition. A commutative ring is a set F equipped with 2 closed associative and commutative operations

$$+ : F \times F \to F$$

$$\times : F \times F \to F$$

(we will write ab for $\times(a, b) = a \times b$), such that:
(1) $\times$ is distributive over $+$:

$$a(b + c) = (ab) + (ac)$$

(2) there is an additive identity element $0 \in F$, so that

$$a + 0 = 0 + a = a \qquad \forall a$$

(3) there is a multiplicative identity element $1 \in F$, so that

$$a1 = 1a = a \qquad \forall a$$

(4) Every $a \in F$ has an additive inverse $-a$ such that $a + (-a) = 0$.

3.34. Example. The integers form a commutative ring.

3.35. Definition. A field is a commutative ring such that

(5) Every $a \in F \setminus \{0\}$ has a multiplicative inverse a^{-1} such that $a(a^{-1}) = 1$.

3.36. Example. The obvious example is the real numbers. The rational numbers also work. As do the complex numbers.

The integers do not work, since 2 has no integer multiplicative inverse.

3.37. The challenge is to find finite sets F that can have all these properties. A great source of such examples comes from thinking about modular arithmetic:
Define a relation of congruence modulo 5 on $\mathbb{Z}$ by $a \cong b$ if $a - b = 5n$ for some integer n.
It is easy to see that this is an equivalence relation. The equivalence classes are:

$$[0] = ..., -10, -5, 0, 5, 10, ...$$

$$[1] = ..., -10+1, -5+1, 0+1, 5+1, 10+1, ...$$

and indeed for $r = 0, 1, 2, 3, 4$:

$$[r] = ..., -10+r, -5+r, 0+r, 5+r, 10+r, ...$$

Miraculously, when we do ordinary integer arithmetic we find that it respects these classes. That is, if $a + b = c$ and a, b are congruent to a', b' respectively then $a' + b'$ is congruent to c. Example:

$$1 + 2 = 3 \qquad 21 + (-98) = -77$$

In this sense we can define arithmetic on the classes mod.p (where at this stage p is any natural number). The resultant structure of integer arithmetic mod.p is denoted $\mathbb{Z}_p$. Thus $\mathbb{Z}_p$ is a set with $+$ and $\times$ which are commutative and associative, distributive...

3.38. Exercise. Check this!

...with additive and multiplicative identity; and additive inverse.

Example: For $p = 5$ the additive inverses of $[0], [1], ...$ are given by

$$[0] + [0] = [0] \qquad [1] + [4] = [0] \qquad [2] + [3] = [0]$$

so that [0]=-[0]; [4]=-[1] and [3]=-[2].

What about multiplicative inverses? Is there an $[x]$ such that $[2][x] = [1]$?

If we are working in $\mathbb{Z}_5$ then: Yes! $[2][3] = [6] = [1]$. And $[4][4] = [16] = [1]$.

Thus

Theorem 3.39. *$\mathbb{Z}_5$ is a field.*

On the other hand $\mathbb{Z}_4$ is a commutative ring, but not a field. The complete row of the multiplication table for [2] is

$$[2][0] = [0] \qquad [2][1] = [2] \qquad [2][2] = [0] \qquad [2][3] = [6] = [2]$$

Since none of the right hand sides is [1] we see that [2] does not have a multiplicative inverse.

In fact

Theorem 3.40. *(i) $\mathbb{Z}_p$ is a field iff p is prime.*
(ii) there is a field of order q iff $q = p^e$ where p is prime and $e \in \mathbb{N}$.
(iii) two fields of the same order are isomorphic.

Part (i) can be proved as an exercise.
Part (ii) is standard in algebra textbooks, but for now we will content ourselves with understanding the statement.
Part (iii) just says that when we have understood part (ii) we will have a handle on *all* finite fields!

So, what about part (ii)? Part (i) tells us how to construct the fields of prime order; and that the fields of order p^2 and so on are *not* $\mathbb{Z}_{p^2}$ and so on.
...so what are they?

One way to address this question is to think about how the rational field sits inside the real field; and the real field inside the complex field. We can ask ourselves what happens when we adjoin $i = \sqrt{-1}$ to $\mathbb{R}$ and try to make a field containing these objects. Since a field is closed under addition we see immediately that the smallest field containing $\mathbb{R}$ and i is $\mathbb{C}$. On the other hand if we adjoin i to $\mathbb{Q}$ we can construct a 'complex rational field' bigger than $\mathbb{Q}$ but smaller than $\mathbb{C}$.

One way of thinking of this is that we have added to $\mathbb{Q}$ a new number v, which number obeys $v^2 + 1 = 0$. We don't really need to know too much else about this number! We can already check the axioms:
A general element of the field can be written in the form $a + bv$ where $a, b \in \mathbb{Q}$. Adding obviously works:

$$(a_1 + b_1 v) + (a_2 + b_2 v) = (a_1 + a_2) + (b_1 + b_2)v$$

and multiplying (using $v^2 = -1$):

$$(a_1 + b_1 v)(a_2 + b_2 v) = (a_1 a_2) + (a_1 b_2 + a_2 b_1)v + (b_1 b_2)v^2$$

$$= ((a_1 a_2) - (b_1 b_2)) + (a_1 b_2 + a_2 b_1)v$$

and the multiplicative inverse is given by $v^{-1} = -v$, since

$$v(-v) = -v^2 = 1$$

and more generally by:

$$Exercise!$$

3.41. Example. What happens if we further extend this field by adding in an object w obeying $w^2 - 2 = 0$?

3.42. The idea for finite fields is to make such extensions to the prime fields $\mathbb{Z}_p$ (p prime). Let us consider the prime $p = 2$, and try to extend the field $\mathbb{Z}_2$. We start by adding in an element that obeys a polynomial equation. We might as well start with a quadratic. Since we want to end up with 'coefficients' in $\mathbb{Z}_2$ the coefficients in the polynomial need to be in $\mathbb{Z}_2$. There is then only one irreducible polynomial available: $f(x) = 1 + x + x^2$. Adjoining a root of f to $\mathbb{Z}_2$ we get a number system consisting of $\{0, 1, x, 1 + x\}$, and that's it! The inverse of x is $1 + x$, since

$$x(1 + x) = x + x^2 = -1 = 1 \qquad (mod.p)$$

This field is called F_4.

More generally we can adjoin a root of an irreducible polynomial of degree e and get F_{2^e}. More generally still, F_{p^e}.

3.5.1 Practical coding with finite fields

We said some time ago that coding is interested in the way the code C sits as a subset in the set of all possible received words (i.e. it is interested in the minimum distance $d(C)$ and so on). From this point of view, the precise choice of symbols used in codewords is not directly relevant. However, realistically, the message itself is quite likely to take the form of strings of letter from some human alphabet — and the recovery of the correct letters at the end of the process is the essential aim. In practice, then, since we are about to start using elements of finite fields to create codes, the question arises: How can we use finite fields to represent our familiar alphabet?

This is the same question as to ask how we can use *any* random set of symbols to represent our alphabet. Doing this is a vital step, if we are going to use new symbol sets. But it is, of itself, essentially trivial. Here we are not trying to maximise Hamming distance or anything like that, so any surjective map from the alphabet to some set of strings of symbols from the new symbol set will do. Thus if we have an alphabet with 26 letters in it (say!), we can represent it with some other symbol set, so long as there are at least 26 codewords available.

3.43. Example. The 26 letters of the alphabet $\{A, B, C, ..., Z\}$ may be represented in $\mathbb{Z}_3^3$ by $A \mapsto 001$, $B \mapsto 002$, $C \mapsto 010$, $D \mapsto 011$, $E \mapsto 012$, ..., $Z \mapsto 222$. This uses up 26 of the $3^3 = 27$ elements of $\mathbb{Z}_3^3$, so we may also represent 'space' by 000.

3.6 Linear codes

Our original idea was to be able to think of S^n as a set of vectors, by making S a field. The analogy was with the case $\mathbb{R}^n$, which is a set of n-component vectors forming a vector space.

If F is a field then F^n is an n-dimensional vector space over F. Addition is component-wise, as usual.

We say that code $C \subset F^n$ is a *linear* code if it is a linear subspace of F^n.

3.44. Example. Let $V = \mathbb{Z}_2^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$. Then $C = \{000, 001, 010, 011\}$ is a subspace.
This is analogous to the fact that $\{(0, y, z) \mid y, z \in \mathbb{R}\}$ is a subspace of the *infinite* space $\mathbb{R}^3$. A basis of $\mathbb{R}^3$ is $\{(0, 0, 1), (0, 1, 0), (1, 0, 0)\}$, and a basis of the subspace is $\{(0, 0, 1), (0, 1, 0)\}$.
A basis of V is $\{(0, 0, 1), (0, 1, 0), (1, 0, 0)\} = \{001, 010, 100\}$, and a basis of C is $\{001, 010\}$.

3.45. Example. Show that if $C, C' \subset F^n$ are linear codes then $C \cap C'$ and $C + C' := \{u + u' \mid u \in C,\ u' \in C'\}$ are also linear codes. When is the code $C \cup C'$ also linear?

3.46. Picking a code at random from $P(F^n)$, it is likely to be non-linear. However "most of the codes currently studied and used are linear" (Jones and Jones, 2000). We will now see why.

When $C \subset F^n$ is linear, and of dimension k as a vector space, then $M = |C| = |F|^k$. We call C a linear $[n, k]$-code.

3.47. The *rate* of a code is

$$R = R(C) = \frac{\log_q M}{n}$$

so for a linear code

$$R = k/n$$

Thus the bigger k is, the more information we transmit; the bigger n is, the longer it takes to transmit. But of course the bigger $n-k$ is the more checking we are doing, so the better we can confirm or protect the information.

Let us now examine some examples of linear codes. In particular, which of the codes we already looked at are linear?

If $S = F$ is a field then the repetition code $R_n \subset F^n$ is linear of dimension 1. Example: 11...1 + 11...1 = 22...2.

The *parity-check* code $P_n \subset F^n$ consists of all vectors u such that

$$\sum_i u_i = 0$$

We can consider the first $n-1$ digits as information, and u_n as a check digit, simply defined as

$$u_n = -\sum_{i=1}^{n-1} u_i.$$

Since it is defined by a linear equation this code is linear. It is a $[n-1, n]$-code, so $M = q^{n-1}$ and $R = n - 1/n$.

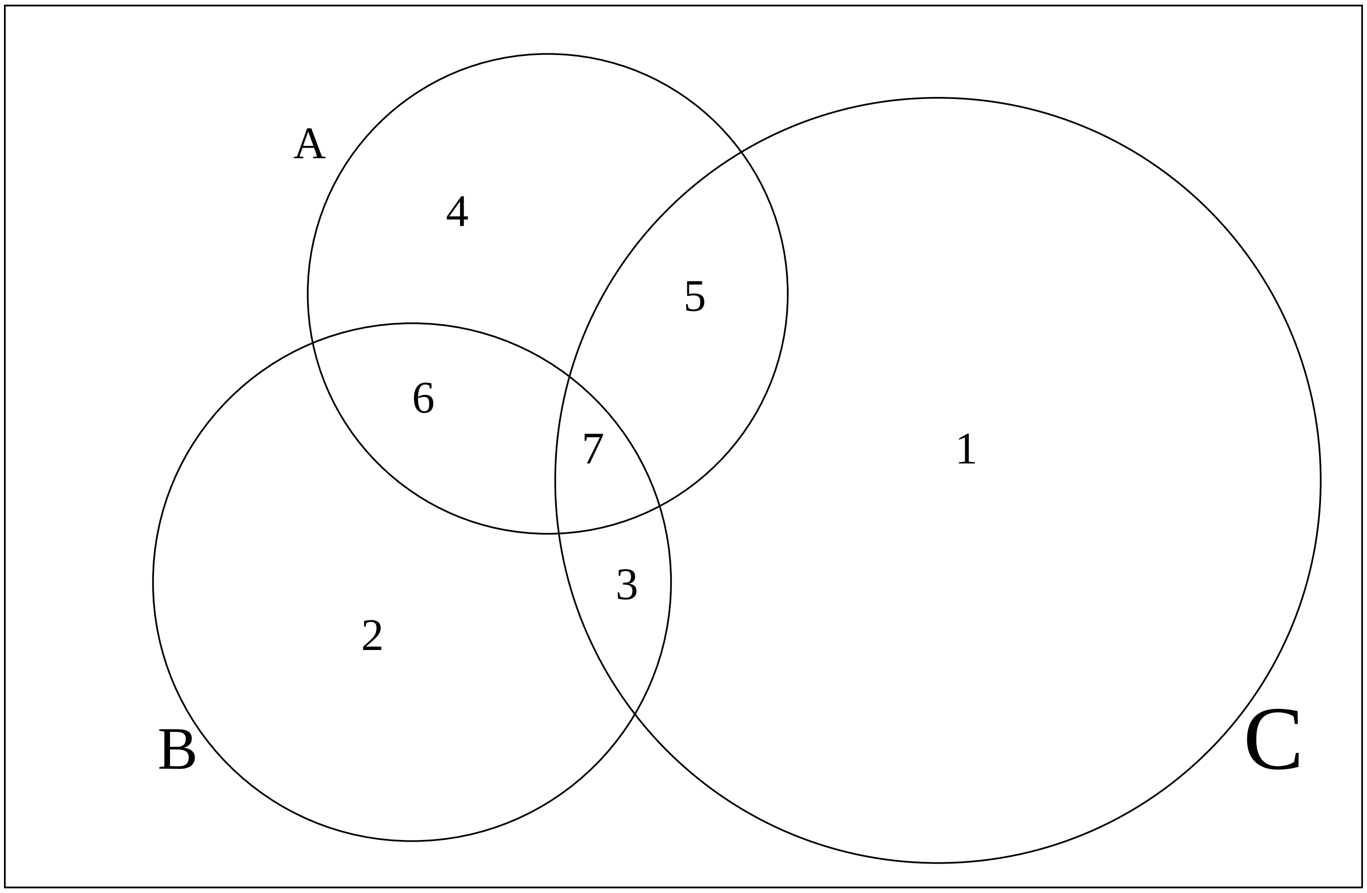


Figure 3.5:

3.48. Example. Consider the Venn diagram for sets A, B, C in Figure 3.5. Suppose we want to encode an element a of $\{0,1\}^4$ as a codeword $u \in S^7$. We will assign the 7 digits to the 7 regions in the figure as numbered. We set $u_3 = a_1$, $u_5 = a_2$, $u_6 = a_3$, and $u_7 = a_4$. We now want to set u_1, u_2, u_4 for collateral (checking) information. We set u_4 so that the sum of digits assigned in set A (i.e. u_4, u_5, u_6, u_7) is zero in binary. We set u_1, u_2 similarly considering C and B.

The code H_7 consists of all codewords $u \in F_2^7$ written in this way.

Since H_7 is determined by linear equations between variables u_i it is a linear code. There are 2^4 choices for a, and these fix u, so $M = 16$. Indeed H_7 has basis $v_1 = 1110000$, $v_2 = 1001100$, $v_3 = 0101010$, $v_4 = 1101001$. Thus the dimension is 4.

We will come back to this example later.

Here are some quick reminders on linear algebra:

3.6.1 Aside on linear algebra

A linear combination of a set of vectors $V = \{v_i\}$ is a form like

$$v = \sum_i a_i v_i$$

Obviously we have

$$0 = \sum_i 0.v_i$$

(on the left we mean the zero vector; on the right the 'scalar/field element/number' 0).

The set of all vectors expressible as linear combinations of V is called the span of V.

3.49. Definition. A set of vectors is linearly independent if the *only* way to linearly combine them to get 0 is with all coefficients 0.

A linearly independent spanning set for a vector space is called a basis.

Theorem 3.50. *Let C be a non-trivial (i.e. non-zero) subspace of V, a vector space over F_q. Then*
(1) C has a basis.
Let $B = \{v_1, v_2, ..., v_k\}$ be a basis for C. Then
(2i) every vector in C can be uniquely expressed as a linear combination in B.
(2ii) $|C| = q^k$.
Proof: Exercise.

Note that any two bases for C have the same order, k. Call this number dim$C$.

3.51. Example. (i) F_q^n has a basis $\{100..00, 010..00, ..., 000..01\}$ consisting of n vectors.
(ii) $C = \{000, 001, 010, 011\}$ is a subspace of $\mathbb{Z}_2^3$ with basis $\{001, 010\}$.
(iii) Is $C = \{000, 001, 002, 010, 020, 011, 022\}$ a subspace of $\mathbb{Z}_3^3$? No! The dim is not a power of 3.

3.52. Proposition. *Let F be a finite field of characteristic p. Then F is itself a vector space over $\mathbb{Z}_p$.*
Proof: Exercise.

3.6.2 Linear codes

3.53. Definition. A q-ary $[n,k,d]$-code is a linear code in F_q^n of dim k and minimum distance d. Write $[n,k,d]-cod$ for the set of all such (with q understood).

Thus $C \in [n,k,d]-cod$ implies $C \in (n,q^k,d)-cod$, but the converse is false.

3.54. Example. Our first three examples are all binary linear codes: $C_1 \in [2,2,1]-cod$; $C_2 \in [3,2,2]-cod$; $C_3 \in [5,2,3]-cod$. Exercise: check this.

Recall that for a general code we need $\frac{1}{2}|C|(|C|-1)$ calculations to compute $d(C)$. We can radically reduce this for a linear code.

To see this first note that

$$d(x,y) = w(x-y)$$

Thus

Theorem 3.55. *For a linear code let*

$$w(C) = min\{w(x) \mid x \in C \setminus \{0\}\}$$

(here we write 0 for the appropriate 000..0 sequence, for convenience). Then

$$w(C) = d(C).$$

Proof: Exercise.

For linear codes we usually just give a basis rather than listing out the whole thing.

3.56. Definition. A $k \times n$ matrix is called a generator matrix for C if its rows form a basis for C.

3.57. Example. C_3 has generator matrix

$$G = \begin{pmatrix} 01101 \\ 10110 \end{pmatrix}$$

However in computing $d(C)$ it is NOT enough to find the minimum weight among the basis vectors! For example

$$G = \begin{pmatrix} 1111 \\ 1110 \end{pmatrix}$$

has min weight 3, but $d(C) = 1$.

Theorem 3.58. *Let G generate C. Any matrix obtained from G by*
(R1) permuting rows
(R2) multiplying a row by a non-zero scalar
(R3) adding one row to another
generates the same code.

3.59. Example. Show that the 3-ary linear codes generated by

$$G = \begin{pmatrix} 210222 \\ 012101 \\ 011112 \end{pmatrix}$$

and

$$G' = \begin{pmatrix} 100201 \\ 010120 \\ 001022 \end{pmatrix}$$

generate the same code. Deduce $d(C)$.
Clues: start by subtracting row two from row three in G. Then subtract row 2 from row 1. Then row 3 from row 2. Then multiply row 1 by the scalar 2. How does it look now?!
Obviously $d(C) \leq 3$, since there is a weight 3 row in G'. But to see if this bound is saturated (it is) you still have some work to do!

3.60. Definition. Codes C, C' are equivalent (write $C \sim C'$) if there is a one-to-one mapping

$$\phi : C \to C'$$

such that

$$d(x, y) = d(\phi(x), \phi(y))$$

for all x, y. In particular $d(C) = d(C')$.

3.61. Exercise. Check $C \sim C'$ is an equivalence relation, i.e. a reflexive, symmetric, transitive relation.

Theorem 3.62. *Let C be a linear code generated by G. Let G' be obtained from G by*
(C1) permuting columns
(C2) multiplying a column by a non-zero scalar $a \in F_q$.
Then G' generates C' an equivalent linear code to C.

Proof: Exercise (optional!).

By using all the row and column operations you can always reduce G to a standard form

$$G' = \left(\begin{array}{c|c} 100..0 & A_{11}A_{12}..A_{1,n-k} \\ 010..0 & A_{21}A_{22}..A_{2,n-k} \\ ... & \\ 000..1 & A_{k1}A_{k2}..A_{k,n-k} \end{array} \right) = [1_k|A]$$

where 1_k is the $k \times k$ unit matrix and A has entries in F_q.

3.63. Example. A binary [5,3,d]-code is generated by

$$G = \begin{pmatrix} 11111 \\ 10011 \\ 11001 \end{pmatrix} \sim \begin{pmatrix} 11111 \\ 01100 \\ 00110 \end{pmatrix} \sim \begin{pmatrix} 10011 \\ 01010 \\ 00110 \end{pmatrix}$$

3.64. Exercise. Let C_i be the 3-ary code generated by G_i, where

$$G_1 = \begin{pmatrix} 1011 \\ 0112 \end{pmatrix}, \qquad G_2 = \begin{pmatrix} 1011 \\ 0111 \end{pmatrix}$$

For each of $i = 1, 2$, list C_i and hence compute $d(C_i)$. Is C_i perfect?

3.6.3 Encoding

Given C, a linear code over F_q (i.e. a subset of F_q^n for some n) generated by G, we have a natural identification between C and F_q^k ($k = dim\ C$, *not* the same as n, the length of the code).
Each $x \in C$ is uniquely expressible as

$$x = \sum_{i=1}^{k} a_i v_i$$

(the v_is are the rows of G in the natural order). So

$$x \leftrightarrow (a_1, a_2, ..., a_k) \in F_q^k$$

is a one-to-one correspondence.

We think of the $a = (a_1, ..., a_k)$ vectors as the *message words* of the code, and the n-tuples x as the codewords representing them.

Note that the encoding map

$$a \rightarrow x$$

is then simply

$$x = aG$$

That is, right multiplication by the generating matrix — a linear map!

3.65. Example. Let C be 3-ary and generated by

$$G = \begin{pmatrix} 10010 \\ 01010 \\ 00102 \end{pmatrix}$$

Encode the messagewords 000, 101 and 122.

Clearly $000 \to 00000$, so we need

$$101 \to (101)G = (10112)$$

$$122 \to (122)G = (12201)$$

Note that the first three digits of the codeword are the same as the messageword. This always happens if G is in the standard form. The other digits are then 'check' digits.

This makes the last part of decoding trivial:

$$\textit{messageword} \overset{encode}{\to} \textit{codeword} \overset{transmit(noise)}{\to} \textit{received vector}$$

$$\overset{project}{\to} \textit{nearest codeword} \overset{interpret}{\to} \textit{decoded messageword}$$

The last step is just to drop off the check digits.

3.6.4 Coset decoding

Our picture above raises a key point. When $x \in F_q^n$ is transmitted down a noisy channel y is received. Define the error vector

$$e = y - x$$

Then the number of transmission errors is $w(e)$. (Of course no one knows both x and y for sure...)
We want an algorithm which decides from y which x was (probably) sent; or equivalently, what e has occurred.

3.66. Definition. Suppose $C \in [n, k, d] - cod_q$ (some d) and $a \in F_q^n$. Then set

$$a + C = \{a + x \mid x \in C\}$$

is called a coset of C in F_q^n.
Theorem 3.67. *[Lagrange] (a) The cosets of a linear code $C \subset F_q^n$ partition F_q^n.*
(b) Each coset has size q^k.
Proof: (Idea) Think of C as a subspace (such as a plane in $\mathbb{R}^3$ through the origin). We can think of a as shifting this subspace parallel-ly away from the plane; in other words to a new plane not including the origin.
We dont have $\mathbb{R}^3$, but the same idea works. $\square$

3.68. Example. Let C be 2-ary generated by

$$G = \begin{pmatrix} 1011 \\ 0101 \end{pmatrix}$$

That is $C = \{0000, 1011, 0101, 1110\}$. Cosets:

$$0000 + C = C$$

$$1000+C = \{1000, 0011, 1101, 0110\} = 0011+C \qquad (etc)$$

and so on.

Given any subset U of F_q^n (such as a coset), we may partition U into subsets containing words of equal weight. Among these will be a subset of words in U of *least* weight. (For example, if we consider the whole of F_q^n then there is always a fixed-weight subset containing just the word of weight zero.)

Henceforth we assume that we always have a way of choosing a single word from any such subset. (If we have totally ordered the words in F_q^n then we could simply take the first one in the order induced on the subset, say.)

Now suppose that the subset we have in mind is a coset. The chosen vector of min weight in a coset is called the *coset leader* (for that choice). E.g. in

$\{0100, 1111, 0001, 1010\}$ either 0100 or 0001 could be chosen as leader.

(**3.6.1**) We can use the idea of coset leaders to generate an arrangement of F_q^n called a *standard array* for C. Assuming as before that we have a way to choose a word from a set (via a total order, say), then we can do this algorithmically:
(i) make a row list of the codewords of C, with 00..0 on the left. This row is coset $00..0 + C$, with 00..0 as coset leader; arranged in some chosen order.
(ii) choose any distinct vector a_1 of min weight in $F_q^n \setminus C$ and row list $a_1 + C$ in the obvious order (i.e. with $a_1 + c$ under codeword c). This has a_1 as coset leader.
(iii) choose any a_2 not already listed, of min weight, and row list $a_2 + C$.
(iv) repeat until all words of F_q^n appear.

Note that there were two kinds of choices in the construction of the standard array: (1)the order in which to write out the row C after 00...0; (2) the choices of coset leaders (the column below 00...0). As we shall see, the first choice has no real bearing on decoding in what follows. The second choice can affect decoding (but all such choices are equally 'good' in probablistic terms).

In our example 3.68 the standard array (for an obvious set of choices) is

$$\left(\begin{array}{c|c|c|c} 0000 & 1011 & 0101 & 1110 \\ 1000 & 0011 & 1101 & 0110 \\ 0100 & 1111 & 0001 & 1010 \\ 0010 & 1001 & 0111 & 1100 \end{array}\right)$$

(don't worry — we won't always need to write out this whole table — see section 3.7.1).

(**3.6.2**) We are now ready to explain coset decoding.

Note that a given standard array A determines, for each word y in F_q^n, a coset leader $e_A(y)$ (the first word in the row of y); and a codeword $c_A(y)$ (the first word in the column of y). For example, the coset leader associated to 1010 in the array above is 0100. Thus if we receive y, we may associate two other words to it, related by

$$e_A(y) = y - c_A(y)$$

Coset decoding:
if we receive y, we decode it as the codeword $c_A(y)$ appearing in the column containing y.

IS this a good strategy?
In coset decoding we are effectively assuming that the

actual error from the transmitted codeword x

$$e = y - x$$

is the coset leader $e_A(y)$ of coset $y + C$.

Suppose the actual error e *is* a coset leader. Then $y = x + e$, so y does lie in the coset with leader e, so $e_A(y) = e$ and $c_A(y) = x$. That is, our decoding is correct.

On the other hand, if the actual error is not a coset leader, then by assuming that it is, we are bound to get the decoding wrong.

By choosing coset leaders to have min weight, we always decode y as the (Hamming) nearest codeword to y (or at least one of the joint nearest). E.g. $y = 0110$ decodes as $x = 1110$ in our example. That is, we assume the fewest errors possible.

In case of low single-digit error probability it is hopefully already clear that this is a good assumption — probablistically. (But see section 3.6.5 for details.)

Returning to our example code, note that $d(C) = w(C) = 2$. Thus it is not even single error correcting, so even single errors might not be corrected properly. (In fact a single error will be corrected if it occurs in the 1st, 2nd or 3rd digit, but not the 4th.)

Specific instances:

messageword $\rightarrow$	*codeword*	*noisy channel*	*decode*	*truncate*	
01	0101	0111 (*say*)	0101	01	(*correct*)
10	1011	1010	1110	11	(*incorrect*)

This glitch is precisely to do with the fact that we had a choice of coset leaders in $0100 + C$. We could have chosen 0001 instead, in which case the 4th digit errors would be recovered and the 2nd digit errors not recovered.

3.6.5 Probability of error correction/detection

As noted before, it is the probabilities of a successful outcome which really dictate the success of our coding methodology. We have accumulated a lot of technology since our last probability calculation, so now it is time to put it all together.

Suppose we transmit a linear code down a symmetric channel with symbol error probability p, then use coset decoding. Then we get the decoding of any received word y right if and only if our error correction is right. This happens in coset decoding if and only if the actual error $e = y - x$ is a coset leader. Thus for any transmitted codeword x

$$P_{corr}(x) = Prob(\text{error } e = \text{one of the coset leaders })$$

(Note that this is independent of x!) In our example 3.68, therefore

$$P_{corr}(x) = P(e = 0000) + P(e = 1000) + P(e = 0100) + P(e = 0010)$$

$$= (1-p)^4 + 3p(1-p)^3$$

$$P_{err}(x) = 1 - P_{corr}(x)$$

Call this $P_{err}(C)$ since it depends only on C, not on x. It is the *word error rate* of the code.

More generally: Let C be any linear code whose coset leaders are $a_0 = 00..0$, $a_1, a_2, ..., a_l$. We have

$$P_{corr}(C) = \sum_{r=0}^{l} P(e = a_r) = \sum_{r=0}^{l} p^{w(a_r)}(1-p)^{n-w(a_r)} = \sum_{s=0}^{n} \gamma_s p^s (1-p)^{n-s}$$

where γ_s is the number of coset leaders of weight s.

How can we compute the γ_ss? In general it is hard, but:

Theorem 3.69. *If $d(C) \geq 2t + 1$ then every vector of weight $\leq t$ is a coset leader for C. Hence*

$$\gamma_s = \binom{n}{s}(q-1)^s = |S_s(00..0)|$$

for $0 \leq s \leq t$. (Recall $S_s(00..0)$ is the sphere around 00..0.)

Proof: Consider the vectors in $B_t(00..0)$. Every vector lies in some coset, so if $y \in B_t(00..0)$ is *not* a coset leader then there exists z with $w(z) \leq w(y)$ and $x \in C$ $(x \neq 0)$ such that

$$y = x + z$$

But then

$$d(C) \leq w(x) = w(y - z) = d(y, z) \leq d(y, 0) + d(0, z)$$

$$= w(y) + w(z) \leq 2w(y) \leq 2t$$

This contradicts the first hypothesis, so y is a coset leader. □

Theorem 3.70. *If C is a perfect $[n, k, 2t+1]$-code then its coset leaders are precisely the vectors of weight $\leq t$.*

3.71. Example. Let C be a 3-ary $[11,6,5]$-code. What is $P_{err}(C)$?

By Theorem 3.69, $d = 5$ implies all vectors of weight ≤ 2 are coset leaders. Therefore

$$\gamma_0 = 1 \qquad \gamma_1 = \binom{11}{1}2^1 = 22 \qquad \gamma_2 = \binom{11}{2}2^2 = 220$$

(For $w > 2$, what is γ_w? We don't know, but let's press on!) Therefore

$$P_{corr}(C) = \sum_{w=0}^{n} \gamma_w p^w (1-p)^{n-w} \geq \sum_{w=0}^{2} \gamma_w p^w (1-p)^{n-w}$$

$$= (1-p)^{11} + 22p(1-p)^{10} + 220p^2(1-p)^9$$

so

$$P_{err}(C) = 1 - P_{corr}(C) \leq 1-((1-p)^{11}+22p(1-p)^{10}+220p^2(1-p)^9)$$

In fact the bound is saturated, because this code is perfect. We know the weights of 1+22+220=243 of the coset leaders. But the number of cosets is

$$\frac{|F_q^n|}{|C|} = q^n/q^k = 3^{11}/3^6 = 3^5 = 243$$

so there *are* no more cosets!

If we use C for error detection, rather than error correction, the analogue of $P_{err}(C)$ is $P_{undetec}(C)$, that is, the probability that a word is received with undetected errors.

Again we transmit $x \in C$ and receive $y \in F_q^n$. The received vector has undetected errors iff $y \neq x$ but $y \in C$. That is, iff $e \in C \setminus \{00..0\}$.

The probability of this is again independent of x:

$$P_{undetec}(C) = \sum_{w=1}^{n} \delta_w p^w (1-p)^{n-w}$$

where δ_w is the number of codewords of weight w.

3.72. Example. $C = \{0000, 1011, 0101, 1110\}$. $\delta_1 = 0$, $\delta_2 = 1$, $\delta_3 = 2$, $\delta_w = 0$ $(w \geq 4)$. Thus

$$\begin{aligned} P_{undetec}(C) &= 0.p(1-p)3 + 1.p^2(1-p)^2 + 2.p^3(1-p) \\ &= p^2(1-p)(1-p+2p) = p^2(1-p^2) \\ &= 0.00009999 \qquad \text{if } p = 0.01 \end{aligned}$$

If y is received and $y \notin C$ we detect an error and request retransmission. How likely is this?

$$\begin{aligned} P_{retrans} &= 1 - P(\textit{no error detected}) \\ &= 1 - (P(\textit{no errors}) + P(\text{error occurs but is not detected})) \\ &= 1 - (1-p)^n - P_{undetec}(C) \end{aligned}$$

In our example

$$P_{retrans}(C) = 0.039394 \qquad \text{if } p = 0.01$$

which is about 4%.

3.7 Dual codes

Notation: Here we use G^t (or G^T) to denote the matrix transpose.
Recall the inner (or scalar) product $u.v$ of two vectors.
3.73. Example. In $\mathbb{Z}_2^4$: 1001.1101=1+0+0+1=0.

3.74. Definition. Given $C \subset F_q^n$, its dual code is

$$C^\perp = \{u \in F_q^n \mid u.v = 0 \; \forall \; v \in C\}$$

3.75. Lemma. *If C generated by G then $v \in C^\perp \iff vG^t = 0$.*

If U, V are vector spaces over F_q and

$$L : U \to V$$

is a linear map, then the range $L(U) \subset V$ is a subspace of V. Its dimension is called the rank of L.
The set of vectors

$$ker\ L = \ \{u \in U \mid L(u) = 0\} \subset U$$

is a subspace of U, called the kernel of U. The dimension of the kernel is called the nullity of L. We have

$$rank\ L + dim(ker\ L) = dim\ U$$

(The Rank-Nullity Theorem from linear algebra.)

Let F be a field (such as F_q). Let us explicitly regard vector space $V = F^n$ as the space of n-component *row* vectors (as has been our convention throughout), i.e. as $1 \times n$ matrices. There is, formally, another realisation as *column* vectors — and even given the choice of row vectors, the explicit matrix representation of individual vectors $v \in V$ depends in principle on a choice of basis. But as soon as we fix all these choices, then each $n \times k$ matrix H with entries in F defines a map

$$L_H : F^n \to F^k$$

by

$$v \mapsto vH$$

Equivalently each $k \times n$ matrix G with entries in F defines a map L^G by $v \to vG^T$.
Let $\alpha, \beta \in F$. Since $H(\alpha v + \beta w) = \alpha Hv + \beta Hw$ we see that L_H is a linear map.
If we consider the standard ordered basis for F^n then its image under L_H will be the set of row vectors in H. Thus

3.76. Lemma. *The dimension of $L_H(F^n)$ (the rank of L_H) is the same as the rank of H* as a matrix.

The same argument holds for L^G and the rank of G.[8]

3.77. Example. Let's try a matrix with rank 1:

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} x+y & x+y \end{pmatrix}$$

Clearly the image space has dim=1 (albeit embedded in a 2d space).

Theorem 3.78. *If C an $[n,k]$-code over F_q (i.e. an $[n,k,d]$-code for some d), then $C^\perp$ is an $[n,n-k]$-code over F_q.*

Proof: Let G be a generator matrix for C, and consider the map

$$L : F_q^n \longrightarrow F_q^k$$

defined by

$$L : v \mapsto vG^T$$

(note that G has k rows and n columns, so G^T has n rows and k columns). Then $C^\perp = ker\ L$ by Lemma 3.75. Thus $C^\perp$ is linear. Now

$$dim\ C^\perp = dim(ker\ L) = dim\ F_q^n - rank\ L = n - rank\ L$$

[8]One can reconstitute all of this for the case where one regards vectors as *column* vectors, simply by 'transposing everything': $v^T \mapsto (vG^T)^T = (G^T)^T v^T = Gv^T$.

But $rank\ L = rank(G) = k$, since a generator matrix has full rank by definition.

□

3.79. Example. $C = \{000, 110, 011, 101\}$ over $\mathbb{Z}_2$ has dimension 2, so the dimension of $C^\perp$ is 3-2=1. We have

$$G = \begin{pmatrix} 110 \\ 011 \end{pmatrix}$$

and $v \in C^\perp$ iff

$$(v_1, v_2, v_3)G^t = (v_1 + v_2, v_2 + v_3) = (0, 0)$$

Over $\mathbb{Z}_2$ this holds iff $v_1 = v_2 = v_3$. Thus $C^\perp = \{000, 111\}$.

Theorem 3.80. *For all linear codes* $(C^\perp)^\perp = C$.

3.81. Definition. Any generator matrix H for $C^\perp$ is called a Parity Check Matrix (PCM) for C.

Theorem 3.80 says $x \in C$ iff $x \in (C^\perp)^\perp$ iff $xH^t = 0$ (via Lemma 3.75). This says that we can think of C as the kernel of the linear map from F_q^n to F_q^{n-k} given by

$$x \mapsto xH^t$$

This says that the $n - k$ rows of H give the coefficients in $n - k$ linear equations which x must satisfy to be a codeword:

$$H_{11}x_1 + H_{12}x_2 + ... + H_{1n}x_n = 0$$

and so on. These are parity check equations (hence PCM).

3.82. Definition. The redundancy of a linear code is $r = n - k$, the number of extra digits added compared to the messageword.

Usually $r < k$ (fewer check digits than message digits), so H is smaller than G and the PCM is a more efficient way to define C than G is.
Given G, can we write down H?...

Theorem 3.83. *Let C be a $[n, k]$-code over F_q generated by*

$$G = [1_k|A]$$

where A is a $k \times (n-k)$ matrix (i.e. G is in standard form).
Then $H = [-A^t|1_{n-k}]$ is PCM for C.
Proof: Exercise.

3.84. Example. 3-ary [6,4]-code generated by

$$G = \left(\begin{array}{cccc|cc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 & 2 \end{array}\right)$$

has PCM

$$H = \left(\begin{array}{cccc|cc} -1 & 0 & -2 & -2 & 1 & 0 \\ -1 & -2 & -1 & -2 & 0 & 1 \end{array}\right) = \begin{pmatrix} 2 & 0 & 1 & 1 & 1 & 0 \\ 2 & 1 & 2 & 1 & 0 & 1 \end{pmatrix}$$

3.85. Definition. A PCM H is in standard form if $H = [B|1_{n-k}]$ where B is a $(n-k) \times k$ matrix.
(Every linear code is equivalent to one whose PCM is in standard form.)

3.7.1 Syndrome decoding

We can use the PCM idea to make decoding more efficient. The idea is, if we receive a vector $y \in F_q^n$ we can compute which coset of C it lies in by computing its syndrome:

3.86. Definition. Let H be a PCM for a $[n, k]$-code C over F_q^n. The syndrome map of C is

$$S : F_q^n \to F_q^{n-k}$$

$$S(y) = yH^t$$

$S(y)$ is the syndrome vector of y. (Note this is a linear map.)

Note that $C = ker\ S$. In fact cosets of C are in 1-to-1 correspondence with syndromes.

3.87. Lemma. *Vectors $u, v \in F_q^n$ are in the same coset of C iff $S(u) = S(v)$.*

Indeed the number of cosets is q^{n-k}, which is the number of vectors in F_q^{n-k}, so cosets and syndromes are in bijective correspondence.

3.88. Example. (NB this is Example 3.68 revisited.) Binary code generated by

$$G = \left(\begin{array}{cc|cc} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{array}\right)$$

gives PCM

$$H = \left(\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right)$$

The coset leaders for C are 0000, 1000, 0100, 0010, and the syndromes: $S(0000) = 00$,

$$S(1000) = (1,0,0,0)\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = 11$$

$$S(0100) = (0,1,0,0)\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = 01$$

$$S(0010) = (0,0,1,0)\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = 10$$

If we receive $y = 1010$ then

$$S(y) = (1010)\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = 01$$

so y is in the coset $0100 + C$. Thus we decode as $x = y - 0100 = 1110$.
Note that we no longer need most of the standard array; just the coset leaders and their syndromes: a syndrome look-up table.

Therefore we have a new decoding scheme:
(i) receive $y \in F_q^n$, calculate $S(y) = z \in F_q^{n-k}$.
(ii) look up z in table, i.e. find the coset leader l (say) such that $S(l) = S(y) = z$.
(iii) decode y as $x = y - l$.
This is much more efficient for large codes.

So, how do we compute $d(C)$ in all this?

Theorem 3.89. *Let C be a [n,k]-code over F_q with PCM H. Then $d(C) = d$ iff every set of $d-1$ columns* *of H is linearly independent, but there exists some set of d columns which is linearly dependent.*

Proof: Let c_i be the i-th column of H. If $x \in C$ has weight w then H has a set of w columns which is linearly dependent: x has w non-zero digits, $x_{i_1}, x_{i_2}, ..., x_{i_w}$ (say), and $xH^t = 0$ so

$$(0, .., x_{i_1}, .., 0, .., x_{i_2}, ..., x_{i_w}, .., 0) \begin{pmatrix} c_1^t \\ c_2^t \\ \vdots \\ c_n^t \end{pmatrix} = \sum_i x_i c_i = 0$$

so the set of w columns $\{c_{i_1}, c_{i_2}, ..., c_{i_w}\}$ is linearly dependent.

Conversely, to each LD set of columns one has a codeword x. If $d(C) = d$ then C has a codeword of weight $w = d$, but no codeword of weight $w = d - 1$. □

Special cases:
$d(C) \geq 2$ iff no set of 1 columns is LD $\iff$ H has no zero columns.
$d(C) \geq 3$ iff no set of 2 columns is LD $\iff$ H has no parallel columns.

3.90. Example. What is $d(C)$ for the binary codes generated by

$$G_1 = \begin{pmatrix} 1011 \\ 0101 \end{pmatrix} \qquad G_2 = \begin{pmatrix} 10110 \\ 01101 \end{pmatrix}$$

giving

$$H_1 = \begin{pmatrix} 1010 \\ 1101 \end{pmatrix} \qquad H_2 = \begin{pmatrix} 11100 \\ 10010 \\ 01001 \end{pmatrix}$$

H_1 has no zero column, but has parallel, so $d(C_1) = 2$; while H_2 has no zero or parallel columns, so $d(C_2) \geq 3$. On the other hand $c_1 + c_3 + c_4 = 0$ for H_2, so $d(C_2) \leq 3$ (since $10110 \in C$). Thus $d(C_2) = 3$.

3.91. Example. Consider linear code C over $\mathbb{Z}_{11}$ with PCM

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & X \end{pmatrix}$$

This C has length 10 (number of columns of H), redundancy 2 (number of rows), so dimension 8. There are no parallel columns, so $d(C) \geq 3$. We have $c_1 - 2c_2 + c_3 = 0$ so $1910000000 \in C$, so $d(C) \leq 3$. Hence $d(C) = 3$ — it is a single error correcting code.

This code has a neat partial decoding scheme:
Since $d(C) = 3$ every vector of weight ≤ 1 is a coset

leader of C (by our earlier result). There are 100 weight 1 vectors in $\mathbb{Z}_{11}^{10}$, namely all non-zero multiples De_i of all e_i (standard ordered basis elements). [9]

The syndrome of the coset led by De_i is given by:

$$S(De_i) = De_iH^t = (0,0,..,0,D,0,..,0)\begin{pmatrix} 11 \\ 12 \\ \vdots \\ 1\,i \\ \vdots \\ 1X \end{pmatrix} = (D, Di)$$

So from (D, Di) we get the coset leader: De_i.

The partial decoding scheme is:

(i) receive $y \in \mathbb{Z}_{11}^{10}$, compute $S(y) = (A, B) \in \mathbb{Z}_{11}^2$.

$$(A,B) = (y_1, y_2, .., y_{10})\begin{pmatrix} 11 \\ 12 \\ \vdots \\ 1\,i \\ \vdots \\ 1X \end{pmatrix} = (\sum_{i=1}^{10} y_i,\ \sum_{i=1}^{10} iy_i)$$

(ii) if $(A, B) = (0, 0)$ then $y \in C$: decode as $x = y$.

(iii) if A, B both nonzero assume single error occurred

[9]There are $11^{10}/11^8 = 121$ cosets altogether.

since (A, B) is $S(De_i)$ for some D, i. Decode as $x = y - Ae_i$ where $i = A^{-1}B$.
(iv) If only one of A, B is non-zero y is not in a coset led by a weight 1 or 0 vector. Therefore at least 2 errors have occured. Request retransmission.

(This is why it is a *partial* scheme. We could have searched through the standard array for weight 2 coset leaders, but they will not be unique, so our 'best' guess will probably have some arbitrariness. Instead just get a retransmission.)

3.92. Example. decode $y = 1025234260 \in \mathbb{Z}_{11}^{10}$:

$$A = \sum_i y_i = 1+2+5+2+3 \;\; + \;\; 4+2+6 = 2+1 = 3$$

$$B = \sum_i iy_i = 1\times1+2\times0+3\times2+4\times5+5\times2+6\times3+7\times4+8\times2+9\times6 = 10$$

We are in case (iii) so assume error is $A = 3$ in digit $i = A^{-1}B = 3^{-1} \times 10 = 4 \times 10 = 7$. Thus subtract 3 from y_7: $x = 1025231260$.
(Exercise: check this is in C!)
On the other hand $y = 2610197034$ has $A = 0$ and $B \neq 0$ (check it!), so seek retransmission in this case.

3.93. Remark. This partial decoding generalises to $d = 2t + 1$ — all vectors of weight $\leq t$ coset leaders:

list their syndromes. If receive y with $S(y)$ in the list decode it; else seek retransmission.

3.7.2 More Exercises

Encoding

Consider linear code C, with generator matrix G. The code is a certain copy of $F_q^k \hookrightarrow F_q^n$ (ideally chosen so that points are Hamming well separated in F_q^n).

So far, we took no account of frequency of use of messagewords, or any other differentiation among messagewords. Thus all points of C, as encodings of messagewords, are of equal standing. In particular there is no reason to try to make some further apart than others. Thus also there is no particular merit in one embedding of the set of messagewords in C over another.

We encode by

$$w \mapsto wG \in C$$

but there are many Gs corresponding to C. Thus for a given message, while fixing C we still get many different encodings.

If $G = G_s$ is in the standard form, we could call the resultant encoding the standard encoding.

If G is a row perm of G_s we might call this semistandard

— the encoding of a message is already different. (This practical change should not be forgotten — noting that the 'code' as we define it has not changed; the PCM does not need to change; and the probablistic effectiveness of the code is not affected.) (An example follows shortly.)
If it is a row and column perm the code changes, and the PCM changes (albeit not in a deep way — the encoding is just permuted by the row perm).
Now read on.

1. The 26 letters of the alphabet may be represented in $\mathbb{Z}_3^3$ by $A \mapsto 001$, $B \mapsto 002$, $C \mapsto 010$, ..., $Z \mapsto 222$. Let us also represent 'space' by 000.
   We are given the parity check matrix

$$H = \begin{pmatrix} 101201 \\ 011100 \\ 000011 \end{pmatrix}$$

   of a linear code C. That is, $w \in C$ iff $Hw^t = 0$. (As usual we write simply 0 for the zero vector, where no ambiguity can arise.)
   For example $H(100012)^t = 0$, so $100012 \in C$.

   (a) Note that H is not in standard form. Confirm that

$$G = \begin{pmatrix} 221000 \\ 120100 \\ 200021 \end{pmatrix}$$

is a generator matrix for C.

ANSWER: This means we have to check that the rows of G are a basis for C. We check (I) that the rows are linearly independent — so that they are a basis for *something*. We confirm this, for example, by noting how the rows differ in columns 3,4 and 6.
(II) that $GH^t = 0$ (by an explicit calculation) — this checks that the rows all *belong* to C.
(III) that the rows span C. Since the dual code has dimension 3 (the number of rows of H) we know that C itself has dimension $6 - 3 = 3$, so G must have 3 rows.

(b) Write down another generator matrix for C. Compute the encoding of the letter E, both by G and by your own choice of alternative generator matrix.

ANSWER: For example

$$G' = \begin{pmatrix} 120100 \\ 221000 \\ 200021 \end{pmatrix}$$

The encoding of E is different by G and by G'. We have

$$(012)G = (012)\begin{pmatrix} 221000 \\ 120100 \\ 200021 \end{pmatrix} = 220112$$

$$(012)G' = (012)\begin{pmatrix} 120100 \\ 221000 \\ 200021 \end{pmatrix} = 021012$$

ASIDE: Note that we do not in general get the messageword from the first digits of the encoded form — this only happens if G is in standard form. Indeed the digits of the messageword might not appear anywhere in the encoded version! This emphasises that the practical encoding of a message depends very much on G, rather than on C.

(c) What is $d(C)$?

ANSWER: Clearly $d(C) \leq 3$, but no column of H is "parallel" to another, so $d(C) = 3$.

(d) How many coset leaders are there? How many coset leaders of weight 1 are there? What are the syndromes of coset leaders?

ANSWER: $|C| = 3^3 = 27$ and $|Z_3^6| = 3^6$ so there are 27 coset leaders. Since $d(C) = 3$ all the weight 1 vectors are coset leaders. There are 12 of these. Their syndromes, and the syndrome $S(000000)$, are easy to compute:

$000000 \mapsto 000$, $x00000 \mapsto x00$ $(x \in \{1, 2\})$, $0x0000 \mapsto 0x0$, $00x000 \mapsto xx0$, $000100 \mapsto 210$, $000200 \mapsto 120$, $0000x0 \mapsto 00x$, $00000x \mapsto x0x$.

The remaining $27 - (12 + 1) = 14$ coset leaders are much harder to find. It is not impossible, since the standard array is not impossibly large in this case, but it is uncomfortable. In practice, a good strategy might be to wait and see what message is *received*, and hence what syndromes we need coset leaders *for* (in order to try to do error correction), rather than just computing them all up front.

Of course there are $(6.5/2)2^2 = 60$ weight 2 vectors in the space. Several, but not all, of these are in cosets led by weight 1 vectors. The syndromes of weight 2 vectors are each easy to compute by linearity, given the weight 1 syndromes above. For example:

$$S(120000) = S(100000)+S(020000) = 100+020 = 120 = S(000200)$$

$$S(010001) = S(010000)+S(000001) = 010+101 = 111 \qquad (\textit{new}!)$$

$$S(001010) = S(001000)+S(000010) = 110+001 = 111 = S(010001)$$

But these cases illustrate the problem. The first is not new; the second is new, and can be taken as a coset leader; but the third is an equally good choice as leader of the same coset (which thus confirms that the code is not reliably 2 error correcting, as we already knew!).

To this point we do not even know if all the remaining coset leaders can be found from among the weight 2 vectors, or whether higher weights are needed. A couple more new ones at weight 2 are: $S(010010) = 011$ and $S(100001) = 201$ (and we can multiply through by 2 to get some more from these), but we would have to keep working through to find the rest. (Exercise!)

This nicely illustrates one of the problems thrown up by coding theory. The syndrome map $S : \mathbb{Z}_3^6 \to \mathbb{Z}_3^3$ is a surjective linear map. The set $\{S(e_1), S(e_2), S(e_5)\}$ is a basis of the image, so we could choose 'coset leaders' of the form $x = \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_5$ with $(\alpha_1, \alpha_2, \alpha_3) \in \mathbb{Z}_3^3$, but this does *not* give the lowest possible weights, so for channels with low single digit error probability this would give highly statistically non-optimal error correction behaviour.

(e) Given that G above is used for encoding, what messageword encodes to 212012, if any? What messageword encodes to 012212, if any?

ANSWER: encoding is

$$(x, y, z) \mapsto (x, y, z)G = (2x+y+2z, 2x+2y, x, y, 2z, z)$$

so for 212012 we could try to solve $2x + y + 2z = 2$, $2x + 2y = 1$, $x = 2$, $y = 0$, $2z = 1$, $z = 2$. The 3-rd, 4-th and 6-th of these give $(x, y, z) = (2, 0, 2)$ (the codeword for the letter T). The others are checks, all of which are satisfied.

For 012212 the 3-rd, 4-th and 6-th of these give $(x, y, z) = (2, 2, 2)$, but two of the checks fail, so 222 is unlikely to be what was intended!

To make a guess for the intended message-word we could compute the syndrome:

$$H(012212)^t = (2, 2, 0)^t$$

The coset leader with this syndrome is 002000. Thus the intended encoding was probably 012212-002000=010212. This decodes as 022=H.

(f) Decode as much as possible of the following received message, given that the transmitted message was encoded using C with generator matrix G, assuming nearest neighbour decoding.

Message:

002112 012212 220112 112100 220112 000000

200021 112000 220112 000000 022022 221000

022200 000000 220112 112000 112000 101200

112000 012020 000000 221000 111112 000000

212012 010212 221000 212021 002000 211121

220112 012021 012021 200021 110221 220112

Hints:

i. The message digits in 212012 are 202 (why?)

ii. 202 is the representation of the 20-th letter: T.

iii. The message digits in 012212 are 222. What is going on here?

ANSWER:
002112H^t = 000 so decode as 212 $->$ W
012212H^t = 220 so must correct by 012212 $\rightarrow$ 012212 $-$ 002000 = 010212 so decode as 022 $\rightarrow$ H
$->$ E $->$ R $->$ E space
A R E
000000 $\rightarrow$ 000 $\rightarrow$ space
022022H^t = 111 so must correct by some choice of weight 2 coset leader (which is at least as likely to be wrong as right, but it no worse than any other choice): choosing 010001 we get 022022 - 010001 = 012021 $->$ 201 $->$ S (choosing 001010 we get 022022 - 001010=021012 $\rightarrow$ K here instead!)
221000H^t = 000 so decode as 100 $\rightarrow$ I
...and so on.

3.8 Hamming codes

Recall: A linear [n,k]-code over F_q with PCM H has $d(C) = d$ iff every set of $d-1$ columns of H is LI, but there is a set of d columns of H that is LD.

For binary codes, no repeated column in H implies $d(C) \geq 3$. Hamming's idea was to construct the biggest possible binary H with no zero columns and no repeated columns. Fixing a positive integer r, then $\mathbb{Z}_2^r$ contains $2^r - 1$ non-zero vectors. We could simply use them all!:

3.94. Definition. Let H be a $r \times (2^r - 1)$ matrix whose columns are the distinct non-zero vectors in $\mathbb{Z}_2^r$. Then $Ham(\mathbb{Z}_2^r)$ is the binary linear code whose PCM is H.

3.95. Example. For $r = 3$:

$$H = \begin{pmatrix} 0001111 \\ 0110011 \\ 1010101 \end{pmatrix}$$

Note columns ordered lexicographically. Really we think of $Ham(\mathbb{Z}_2^r)$ as a collection of several different equivalent codes, since we can order the columns as we like.

For $Ham(\mathbb{Z}_2^3)$ we could write

$$\tilde{H} = \left(\begin{array}{c|c} 1110 & 100 \\ 1101 & 010 \\ 1011 & 001 \end{array}\right)$$

which is in the standard form. Then the generator matrix is

$$\tilde{G} = \left(\begin{array}{c|c} 1000 & 111 \\ 0100 & 110 \\ 0010 & 101 \\ 0001 & 011 \end{array}\right)$$

3.96. Exercise. Connect this formulation to the example introduced earlier.

Theorem 3.97. *$Ham(\mathbb{Z}_2^r)$ has minimum distance 3 and is perfect.*

Proof: H has no zero or parallel columns by construction, so $d \geq 3$. But it contains columns c_1, c_2, c_3 in lex order obeying $c_1 + c_2 + c_3 = 0$, so $d = 3$. Hence $Ham(\mathbb{Z}_2^r)$ is perfect iff the collection of 1-balls centred on codewords exhausts $\mathbb{Z}_2^n$, where $n = 2^r - 1$. But

$$|B_1(x)| = 1 + \binom{n}{1} = 1 + n = 2^r$$

and $M = |Ham(\mathbb{Z}_2^r)| = 2^k$ where $k = 2^r - 1 - r$. So

$$|\sqcup_{x \in Ham(\mathbb{Z}_2^r)} B_1(x)| = 2^k \times 2^r = 2^n$$

□

Hence the coset leaders of $Ham(\mathbb{Z}_2^r)$ are all vectors of weigth ≤ 1. Note that weight 1 binary vectors are just the e_is. Syndrome:

$$S(e_i) = e_i H^t = (0,0,..,0,1,0,..,0) \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix} = c_i$$

(here we write c_i for the columns written out as rows, for brevity).

If the columns are ordered lexicographically then the i-th column is just the binary representation of i. So if we receive $y \in \mathbb{Z}_2^n$ with one error, its syndrome $S(y)$ is the digit position of the error (in binary).

3.98. Example. receive $y = 1101101$. Then

$$S(y) = (1,1,0,1,1,0,1)H^t = 101 = S(e_5)$$

Syndrome decoding: $x = y - e_5 = 1101001$.

3.8.1 Hamming over non-binary fields

3.99. Definition. Let $u, v \in F_q^r \setminus \{0\}$. u is projectively equivalent to v, written $u \sim v$, if there exists $\lambda \in F_q \setminus \{0\}$

such that $u = \lambda v$.

This says that u, v are parallel. (NB, being parallel is an equivalence relation.)

We call the set of projective equivalence classes the projective space of F_q^n, denoted $P(F_q^n)$.

3.100. Example. $\mathbb{Z}_5^2$ has the following projective equivalence classes:

$$[01] = \{01, 02, 03, 04\}$$

$$[10] = \{10, 20, 30, 40\}$$

$$[11] = \{11, 22, 33, 44\}$$

$$[12] = \{12, 24, 31, 43\}$$

...

$$[14] = \{14, 23, 32, 41\}$$

In general there are $q - 1$ elements in each class, so there are $\frac{q^r-1}{q-1}$ projective equivalence classes.

3.101. Definition. Let H be a $r \times \frac{q^r-1}{q-1}$ matrix (over F_q) each of whose columns belongs to a different class in $P(F_q^r)$. Then the q-ary linear code whose PCM is H is a q-ary Hamming code, denoted $Ham(F_q^r)$.

3.102. Example. For $Ham(F_5^2)$ we could choose PCM

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 & 4 \end{pmatrix} \qquad \mathit{or} \qquad H = \begin{pmatrix} 0 & 3 & 4 & 1 & 2 & 3 \\ 2 & 0 & 4 & 2 & 1 & 2 \end{pmatrix}$$

Of these, H is best for easy decoding in practice. In H we chose from each class the unique vector whose first non-zero digit is 1; and then ordered the vectors lexicographically. (If we refer to *the* code $Ham(F_5^2)$, this is the PCM we mean.)

Theorem 3.103. *$Ham(F_q^r)$ has minimum distance 3 and is perfect.*

Proof: Exercise (optional).

SYNDROME DECODING:

Again we know that coset leaders are vectors of weight ≤ 1, that is, the zero vector (let's call it $\underline{0}$); and the vectors of form Ae_i, where $A \in F_q \setminus \{0\}$ and $1 \leq i \leq n$.

Syndromes: $S(\underline{0}) = \underline{0}$

$$S(Ae_i) = Ae_iH^T = A[0,0,..,0,1,0,..,0]\begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_i \\ \vdots \\ c_n \end{pmatrix} = Ac_i$$

NB, if H is our 'standard' choice , then first non-zero digit of c_i is 1 implies first non-zero digit of $S(Ae_i)$ is A, which implies that we can read off A immediately.

SCHEME: (i) receive y; compute $S(y)$;

(ii) If $S(y) = 0$ then $x = y$;

(iii) any other $S(y) \in F_q^r$ must lie in one of the classes of $P(F_q^r)$, so $S(y) = Ac_i = S(Ae_i)$ for some $A \in F_q \setminus \{0\}$, $1 \leq i \leq n$.

Decode by subtracting A from digit i:

$$y \mapsto x = y - Ae_i$$

3.104. Example. $Ham(F_4^2)$

$n = |P(F_4^2)| = \frac{4^2-1}{4-1} = 5$, $r = 2$ implies $k = 3$, so we have a $[5, 3, 3]$-code over F_4.

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & a & b \end{pmatrix}$$

Suppose we receive $y = bab10$. We have

$$S(y) = [b, a, b, 1, 0]H^T = [a+b+1+0, b+b+a] = [1+1, a] = [0, a] = a[0, 1]$$

so

$$y \mapsto x = y - ae_i = [b - a, a, b, 1, 0] = 1ab10.$$

In summary, this is very similar to previous examples. The main change is in the type of arithmetic done.

3.9 Cyclic codes

We start this section by introducing the technology we shall need. The use we shall make of it comes later.

3.105. Definition. A code C is cyclic if it is linear and any cyclic shift of a codeword is also a codeword.

3.106. Example. 2-ary code $C = \{000, 101, 011, 110\}$ is cyclic.

3.9.1 Some rings and fields

3.107. Definition. Let F be a field. Then $F[x]$ is the set of all polynomials in x:

$$a(x) = \sum_i a_i x^i$$

where $a_i \in F$. If $a(x)$ has degree m and $a_m = 1$ then $a(x)$ is said to be monic.

$F[x]$ is a ring, but not a field.

Associated to any polynomial $a(x) \in F[x]$ there is a function: $x \mapsto a(x)$ (the evaluation function). In general a polynomial is more than a function, however, in the following sense.

3.108. Example. There are 4 distinct functions from $\mathbb{Z}_2 \to \mathbb{Z}_2$. But there are infinitely many different polynomials in $\mathbb{Z}_2[x]$. E.g. $a(x) = x^5+x^2+x+1$, $b(x) = x^{17}+1$, both have the same function associated to them (exercise!).

Theorem 3.109. *[The remainder theorem]*
For every pair $a(x), b(x) \in F[x]$ with $b(x) \neq 0$, there exists a unique pair $q(x)$ (the quotient) and $r(x)$ (the remainder) in $F[x]$ such that $deg(r(x)) < deg(b(x))$ and $a(x) = q(x)b(x) + r(x)$.

Proof: Can construct $q(x)$, $r(x)$ by usual long-division algorithm, using appropriate arithmetic. □

3.110. Exercise. Divide $a(x) = x^3 + 3x^2 + 4$ by $b(x) = 2x^2 + 3$ in $\mathbb{Z}_5[x]$. [10]

3.111. Definition. Choose a *fixed* polynomial $f(x) \in F[x]$. Then polynomials $a(x), b(x) \in F[x]$ are *congruent modulo* $f(x)$ (written $a(x) \equiv b(x)$ mod. $f(x)$), if $a(x) - b(x)$ is divisible by $f(x)$ (meaning $a(x)-b(x) = q(x)f(x)$

[10] Answer:

$$x^3 + 3x^2 + 4 = (3x + 4)(2x^2 + 3) + (x + 2)$$

for some $q(x) \in F[x]$, with no remainder).

This is an *equivalence relation* on $F[x]$ (check it! This is just like our modular arithmetic).

As usual, denote equivalence (congruence) class of $a(x)$ by

$$[a(x)] = \{b(x) \in F[x] \mid b(x) \equiv a(x) \ \ mod. f(x)\}$$

Let $F[x]/f(x)$ denote the set of such classes. We can define addition and multiplication on $F[x]/f(x)$:

$$[a(x)] + [b(x)] = [a(x) + b(x)]$$

$$[a(x)][b(x)] = [a(x)b(x)]$$

(These are well defined by a lemma that you should state and check, analogous to one we had earlier.)

By these operations $F[x]/f(x)$ is a ring.

Any polynomial $a(x) \in F[x]$ has a unique remainder $r(x)$ 'modulo' $f(x)$, with $deg(r(x)) < deg(f(x))$ by Theorem 3.109.

3.112. Lemma. *$a(x) \equiv a'(x)$ mod. $f(x)$ iff their remainders $r(x), r'(x)$ are equal.*

The upshot of this is that we can identify $[a(x)]$ with $r(x)$, the remainder of any of its elements. In this way we may identify

$$F[x]/f(x) \leftrightarrow \{\sum_{i=0}^{n-1} a_i x^i \mid a_0, a_1, .., a_{n-1} \in F\}$$

the set of polynomials of degree $< \ deg(f(x)) = n$. Of course, this set may then be identified with F^n — the list of coefficients.

Altogether this gives us a way to regard the vector space F^n as a ring. That is, we equip it with the extra operation of multiplication of vectors!

3.113. Example. $R = \mathbb{Z}_2[x]/(x^2+x+1)$ (NB $f(x)$ here has degree 2), gives $R \equiv \{0, 1, x, 1+x\}$ = polynomials of degree < 2.

Compute the addition and multiplication tables.

Can you compute inverses too?

In fact every non-zero element does have an inverse, so R is even a field in this case!

3.114. Definition. $f(x) \in F[x]$ is reducible if there exist $a(x), b(x) \in F[x]$ with degrees less than that of $f(x)$, such that $f(x) = a(x)b(x)$.

FACT: $F[x]/f(x)$ is a field iff $f(x)$ is not reducible (irreducible).

3.115. Lemma. *(i) $f(x) \in F[x]$ has a degree 1 factor $(x-a)$ iff $f(a) = 0$.*

(ii) If degree $f(x)$ =2 or 3 then $f(x)$ is irreducible iff for all $a \in F$, $f(a) \neq 0$.

(iii) Over any field F, $x^n - 1 = (x-1)(x^{n-1} + x^{n-2} + ... + x + 1)$.

Proof: (i) Use Theorem 3.109. (iii) by induction on n.
□

3.116. Example. Completely factorise $x^4 - 1 \in \mathbb{Z}_5[x]$.
[11]

For cyclic codes the ring of interest is as follows.

3.117. Definition. For given field F, define

$$R_n = F[x]/(x^n - 1)$$

NOTES:
(a) $(x^n - 1)$ is always reducible, so R_n is never a field.
(b) $x^n \equiv 1$ mod. $x^n - 1$, so $x^{n+m} = x^m$ for any m. No need to use remainder theorem to compute products.
E.g. in $R_5 = \mathbb{Z}_3[x]/(x^5 - 1)$

$$(x^2+x)(x^4+2) = x^6+2x^2+x^5+2x \equiv x+2x^2+1+2x = 2x^2+1$$

(c) Since $deg(x^n - 1) = n$ we can identify R_n with polynomials of degree less than n, and hence with F^n:
$a_0 + a_1x + ... + a_{n-1}x^{n-1} \leftrightarrow (a_0, a_1, .., a_{n-1})$
addition of polys $\leftrightarrow$ vector addition

[11] Answer: over $\mathbb{Z}_5$

$$x^4 - 1 = (x-1)(x-2)(x-3)(x+1) = (x+4)(x+3)(x+2)(x+1)$$

multiplication by constant $\leftrightarrow$ scalar multiplication

multiplication by $x \leftrightarrow$ cyclic shift.

3.9.2 Back to the codes

We can think of a q-ary code of block-length n as a subset of R_n (with $F = F_q$). Then:

Theorem 3.118. *A code $C \subset R_n$ is a cyclic code iff*
(i) $a(x), b(x) \in C$ implies $a(x) + b(x) \in C$;
(ii) $a(x) \in C$, $r(x) \in R_n$ implies $r(x)a(x) \in C$.
(NB (ii) is more than closure of C under multiplication!)

3.119. Definition. Let $f(x) \in R_n$. Then

$$\langle f(x)\rangle \;=\; \{r(x)f(x) \mid r(x) \in R_n\}$$

called "the ring span of $f(x)$".
Clearly this satisfies properties (i), (ii) of Theorem 3.118. Hence it is a cyclic code over F_q — the cyclic code generated by $f(x)$.

3.120. Example. $F_q = \mathbb{Z}_2$, $C = \langle 1 + x^2\rangle \subset R_3 = \mathbb{Z}_2[x]/(x^3 - 1)$

$$R_3 = \{0, 1, x, 1 + x, x^2, 1 + x^2, x + x^2, 1 + x + x^2\}$$

$$C = \{0, 1{+}x^2, x{+}1, x^2{+}x, 1{+}x, x{+}x^2{+}x^3{+}x^4, 1{+}x{+}x^2{+}x^2{+}x^3{+}x^4\}$$

$= \{0, 1+x^2, 1+x, x+x^2\} \leftrightarrow \{000, 101, 110, 011\} \subset \mathbb{Z}_2^3$

3.121. Exercise. Show that $\langle 1+x^2 \rangle = \langle 1+x \rangle = \langle x+x^2 \rangle$ in this case.

That is, more than one polynomial can generate a given cyclic code. However, there is a canonical choice of generating polynomial:

Theorem 3.122. *Let C be a non-zero cyclic code in R_n. Then*
(i) there exists a unique monic polynomial $g(x)$ of least degree in C;
(ii) $C = \langle g(x) \rangle$. In fact every codeword $a(x) \in C$ is a strict *multiple of $g(x)$: $a(x) = r(x)g(x)$ (not just congruent mod. $x^n - 1$).*
(iii) $g(x)$ is a factor of $x^n - 1$.

3.123. Definition. The unique minimal degree monic polynomial $g(x)$ in a cyclic code C is called the generator polynomial of C.

For example, $g(x) = 1 + x$ is the gen. poly. for our last example.

CRUCIAL FACT:
Since the gen. poly. is unique, cyclic codes of length n are in 1-to-1 correspondence with monic factors of $x^n - 1$.

This completely characterises all cyclic codes!

3.124. Example. (a) Find all cyclic codes over $\mathbb{Z}_2$ of length 3.
$C \subset R_3 = \mathbb{Z}_2[x]/(x^3-1)$. But

$$x^3 - 1 = (x-1)(x^2+x+1) = (x+1)(x^2+x+1)$$

so there are four such codes: $\langle 1\rangle = R_3 = \mathbb{Z}_2^3$
$\langle 1+x\rangle = \{0, 1+x, 1+x^2, x+x^2\} = \{000, 110, 101, 011\}$
$\langle 1+x+x^2\rangle = \{0, 1+x+x^2\} = \{000, 111\}$
$\langle (1+x)(1+x+x^2)\rangle = \{0\} = \{000\}$
(b) How many cyclic codes of length 4 over $\mathbb{Z}_5$ are there?
Answer: same as number of monic factors of $x^4-1 \in \mathbb{Z}_5[x]$. But we already saw that $x^4 - 1 = (x+4)(x+3)(x+2)(x+1)$ over $\mathbb{Z}_5$, so the general monic factor is

$$g(x) = (x+4)^{p_4}(x+3)^{p_3}(x+2)^{p_2}(x+1)^{p_1}$$

where each p_i can be either 0 or 1. Since there are 2^4 choices here, we have 16 cyclic codes.
For example $p_1 = p_4 = 1$, $p_2 = p_3 = 0$ gives code $\langle (x+1)(x+4)\rangle = \langle x^2-1\rangle$

What can we say about this code? What is its dimension?

Theorem 3.125. *Let $g(x) = \sum_{i=0}^{r} g_i x^i$ be the gen. poly.*

for cyclic code C (note $g_r = 1$). Then

$$G = \begin{pmatrix} g_0 & g_1 & g_2 & .. & g_r & 0 & .. & 0 \\ 0 & g_0 & g_1 & g_2 & .. & g_r & .. & 0 \\ & & \ddots & & & & & \\ 0 & ... & 0 & g_0 & g_1 & g_2 & .. & g_r \end{pmatrix}$$

is a generator matrix for C.

3.126. Corollary. *A cyclic code $C \subset R_n$ whose gen. poly. has dim. $k = n - r$ has redundancy r.*

3.127. Example. Construct a generator matrix for each 3-ary cyclic code of length 4.

$R_4 = \mathbb{Z}_3[x]/(x^4 - 1)$

$$(x^4-1) = (x^2-1)(x^2+1) = (x-1)(x+1)(x^2+1) = (x+1)(x+2)(x^2+1)$$

(NB $(x^2 + 1)$ is irreducible here) so there are 2^3 monic factors, hence 8 cyclic codes generated by

$$g(x) = (x + 1)^{p_1}(x + 2)^{p_2}(x^2 + 1)^{p_3}$$

with $p_i \in \{0, 1\}$. We have the list given in Table 3.2.

To find $d(C)$, do syndrome decoding, etc, it is better to have a PCM than a generator matrix. So how can we construct H here?

Recall H is generator matrix for $C^\perp$.

Theorem 3.128. *If C is cyclic so is $C^\perp$.*

$g(x)$	*redundancy*	*dimension*	G
1	0	4	1_4
$1+x$	1	3	$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$
$2+x$	1	3	$\begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \end{pmatrix}$
$1+x^2$	2	2	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$
$(1+x)(2+x)$	2	2	$\begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{pmatrix}$
$(1+x)(1+x^2)$	3	1	$\begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}$
$(2+x)(1+x^2)$	3	1	$\begin{pmatrix} 2 & 1 & 2 & 1 \end{pmatrix}$
$(1+x)(2+x)(1+x^2)$	4	0	$-$

Table 3.2:

So $C^\perp$ has a unique generator polyomial, that is also a factor of $x^n - 1$. If we find it we can us Theorem 3.125 to find a generator matrix for $C^\perp$ and hence a PCM for C.

3.129. Definition. Let $C \subset R_n$ be cyclic with generator polynomial $g(x)$. Then Theorem 3.122 implies that there exists another polynomial $h(x)$ such that $x^n - 1 = g(x)h(x)$, and $h(x)$ is unique by Theorem 3.109. We call $h(x)$ the *check polynomial* of the code C.

3.130. Example. Given that $g(x) = x^2 + x + 3$ is the gen. poly. of a cyclic 5-ary [4,2]-code C, we have

$$(x^2 + x + 3)(x^2 + 4x + 3) \equiv x^4 - 1$$

so $h(x) = (x^2 + 4x + 3)$.
Note incidentally that $C^\perp \neq \langle h(x) \rangle$.

Theorem 3.131. *Let $h(x)$ be the check poly. for code C. Then $a(x) \in C$ iff $a(x)h(x) \equiv 0$.*

It is not true in general that $C^\perp = \langle h(x) \rangle$, but we can construct the gen. poly. for $C^\perp$ from $h(x)$.

Define

$$H = \begin{pmatrix} h_k & h_{k-1} & h_{k-2} & .. & h_0 & 0 & .. & 0 \\ 0 & h_k & h_{k-1} & h_{k-2} & .. & h_0 & .. & 0 \\ & & \ddots & & & & & \\ 0 & ... & 0 & h_k & h_{k-1} & h_{k-2} & .. & h_0 \end{pmatrix}$$

We have

$$\begin{pmatrix} a_0 & a_1 & \dots & a_{k-1} \end{pmatrix} H = 0 \tag{3.3}$$

for $a(x) \in C$.

Consider C', code generated by H. Since $h(x)$ is monic we have $h_k = 1$, so leading diagonal is all 1s. Thus H has maximal rank $((n-k))$, so $dim\ C' = n-k$. Also, any $\underline{w} \in C'$ is perpendicular to all $a(x) \in C$ by (3.3). Thus $C' \subset C^{\perp}$. But $dim\ C' = dim\ C^{\perp}$, so $C' = C^{\perp}$.

Theorem 3.132. *Let $C \subset R_n$ be a cyclic code with check poly. $h(x)$. Then H is a PCM for C.*

3.133. Example. Recall $h(x) = 3 + 4x + x^2$ is check poly. for $C = \langle x^2 + x + 3 \rangle \subset R_4 = \mathbb{Z}_5[x]/(x^4 - 1)$. Hence

$$H = \begin{pmatrix} 1 & 4 & 3 & 0 \\ 0 & 1 & 4 & 3 \end{pmatrix}$$

is a PCM for C.

Exercise: check $aH^T = 0$ for all $a \in C$.

THIS is what we want! A construction for the PCM for C. Armed with this, we can do our usual routines for coding with C.

It remains to compute a gen poly. for $C^{\perp}$:

Comparing G and H we see that they are of similar form. However the reversing of the indices means that it is an open question whether

$$g^{\perp}(x) \ = h_k + h_{k-1}x + ... + h_0 x^k$$

— the candidate for the gen. poly. for $C^{\perp}$ on this basis — is monic.
We can obtain a monic version by dividing by h_0... ...unless $h_0 = 0$.
But we need not worry about this: If $h_0 = 0$ then $h(0) = 0$ and

$$x^n - 1 = g(x)h(x) \quad \Rightarrow \qquad -1 = g(0)h(0) = 0$$

which cannot happen.

3.134. Definition. Given $p(x) = p_0 + p_1 x + ... + p_k x^k \in F_q[x]$ ($p_k \neq 0$) the *reciprocal* of $p(x)$ is

$$\overline{p}(x) = p_k + p_{k-1}x + ... + p_0 x^k \in F_q[x]$$

So we have

3.135. Corollary. *Let $C \subset R_n$ be a cyclic code with check poly. $h(x)$. Then $C^{\perp}$ is the cyclic code with gen poly.*

$$g^{\perp}(x) = h(0)^{-1}\overline{h}(x)$$

3.136. Example. $C = \langle x^2+x+3 \rangle \subset R_4 = \mathbb{Z}_5[x]/(x^4 - 1)$ has check poly $h(x) = 3 + 4x + x^2$. Hence gen poly. for $C^\perp$ is

$$g^\perp = h(0)^{-1}\overline{h}(x) = 3^{-1}(1+4x+3x^2) = 2(1+4x+3x^2) = 2+3x+x^2$$

3.10 More Exercises

3.137. Exercise. Can you describe an 'alphabet' Σ_q of size q, and give an n such that this entire *question* is a codeword in some $C \subset \Sigma_q^n$?
Answer:

If we have a Σ_q consisting of all upper and lower case letters, all Greek letters, all punctation, some typesetting instructions (subscript etc), and a 'space' symbol, then we can assemble the question from these. The q is roughly 52+20+20+20 (say). Let's add in some math symbols too, and say $q = 130$.
For n we just count up the number of symbols in the question, including spaces etc: roughly $n = 100$.
Another construction would be to have a Σ_q containing highly complex composite symbols, whose shapes form whole words, or perhaps even whole sentences. Of course

this is much less realistic (the symbol set would be difficult or impossible to use in most circumstances), but in the extreme we could have $q = 1$ (the element is an image of the whole question) and $n = 1$!

3.138. Exercise. A code C is known to be 21 error correcting. State a lower bound on $d(C)$.
Answer:

By our Proposition: If $d(C) \geq 2t + 1$ then C can correct up to t errors by the 'pick the closest' strategy. Thus in our case $d(C) \geq 43$.

3.139. Exercise. The set E_n of even weight binary vectors of length n is a subspace of $\mathbb{Z}_2^n$. Hence E_n is a binary linear code. What are the parameters $[n, k, d]$ of E_n? Write down a generator matrix for E_n in standard form.
Answer:
$k = dim\ E_n = n - 1$.
d is the minimum weight of non-zero vectors in E_n , which must be 2 (all vectors have even weight, so $d \geq 2$, and $1100 \in E_n$ has weight 2). Hence E_n is a binary linear $[n, n-1, 2]$-code. Its generator matrix in standard

form is

$$\begin{pmatrix} 100...01 \\ 010...01 \\ 001...01 \\ .... \\ 000...11 \end{pmatrix}$$

3.140. Exercise. (i) Construct a binary linear [8,4,3]-code.
(ii) How many different matrices in standard form generate such codes?
Answer:

(i) We are looking for a length 8 code $C \subset \mathbb{Z}_2^8$, with dimension 4, thus we are looking for a 4×8 generator matrix. Putting this in standard form (without loss of generality) means

$$G = [1_4|A]$$

where A is a 4×4 binary matrix. There are a grand total of $2^{4\times 4}$ such matrices, including the zero matrix; the identity matrix and so on. Not all of them generate $d = 3$ codes, however. For example $A = 0$ means that each generating vector (in $G = [1_4|0]$) has total weight 1 (so $d = 1$). Similarly $A = 1_4$ means that each generating vector (in $G = [1_4|1_4]$) has total weight 2.

In other words each row of A must have at least two non-zero entries (so that each row of G has at least three), if we want $d = 3$. For example

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

Now for each candidate, such as this, we need to check that the minimum weight of all nonzero vectors is 3. There are several ways to do this. One way is to construct the PCM:

$$H = [-A^t|1_4] = [A^t|1_4]$$

Here we just need to check that no column is zero and no two columns are the same (by Theorem 3.89). This is clearly the same as checking that no two *rows* of A are the same (we have stipulated that each row has at least two non-zero entries, so their transposes cannot be the same as any of the vectors in 1_4).

This requirement is satisfied in our example, so we are done. The code in full consists in all linear combinations of the row vectors in our G. Thus it starts

$$C = \{00000000, 10001100, 01001010, 00101001, 00010011, 11000110, 101$$

$$..., 11111100\}$$

(16 elements altogether).

Remark: Another example satisfying the criteria is give by

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

Remark: $A_2(8,3) = 20$, so a binary (8,20,3)-code exists. Of course no such code can be linear, since $|C| = M = q^k$ for some integer dimension k for a linear code, and no integer k obeys $20 = 2^k$! Thus $|C| = 2^4 = 16$ is the biggest *linear* code we can hope for.

(ii) As for the number of such distinct generating matrices $G = [1_4|A]$, we can choose the first row of A freely, except not choosing 0000, or any of the four weight-1 vectors, so there are $2^4 - 5$ possibilities. The second row can be chosen freely except not 0000, or weight-1, and not the same as the first row, so there are $2^4 - 6$ possibilities. Continuing similarly, altogether we have $\frac{(2^4-5)!}{(2^4-9)!}$ choices.

3.141. Exercise. (i) Construct standard arrays for the binary linear codes C_1, C_2 generated by

$$G_1 = \begin{pmatrix} 101 \\ 011 \end{pmatrix} \qquad G_1 = \begin{pmatrix} 10110 \\ 01011 \end{pmatrix}$$

(ii) Decode the received vectors 11111 and 11011 and 01011 in C_2.
(iii) We receive 00101. What is going on?! Explain with a ball-packing analogy.

Answer:
(i) $C_1 = \{000, 101, 011, 110\}$. Evidently this has $d(C_1) = 2$.
As usual, the first row of the array is

$$000 \;\; 101 \;\; 011 \;\; 110$$

Since 100 is not in the code, and hence has not appeared in the array so far, we can use it as the next coset leader (we could have used 001 instead, say). We get the next row:

$$100 \;\; 001 \;\; 111 \;\; 010$$

(just vector shift all the code elements by 100).
At this point we see that all eight vectors in $\mathbb{Z}_2^3$ are IN, so we stop.
Since the weight 1 vectors are not in distinct rows, this

code is not even 1 error correcting. (Of course we knew that already, since $d(C_1) < 3$.)

For C_2 we start with the code itself in the first row:

$$00000 \;\; 10110 \;\; 01011 \;\; 11101$$

Then we construct rows with coset leaders which are (a) not in the code; (b) of lowest possible weight, i.e. of weight 1. Since 10000 has not appeared in the code we lead with that next:

$$10000 \;\; 00110 \;\; 11011 \;\; 01101$$

then

$$01000 \;\; 11110 \;\; 00011 \;\; 10101$$

In this case none of the weight 1 vectors appear in each other's cosets, so we construct a total of 5 rows this way, continuing with coset leaders 00100, 00010 and 00001. E.g.

$$00010 \;\; 10100 \;\; 01001 \;\; 11111$$

Since $|\mathbb{Z}_2^5| = 32$ and each row has 4 vectors in it, we need 8 rows altogether. We have 6 so far. The remainder will have to be led by vectors of higher weight.

Some weight 2 vectors have already appeared, but 11000, 10001, 00101, 01100 have not. We can make:

11000 01110 10011 00101

and

10001 00111 11010 01100

and then we are done.
Of course we could have started with one of the others, which would have produced a different array! This tells us that our code is not reliably 2 error correcting.
(ii) 11111 lives in a column below codeword 11101
11011 lives in a column below codeword 01011
01011 is a codeword.
(iii) We decode 00101 as 11101, because we decide that the error in this case was 11000, based on our array (11000 is the coset leader). Clearly there is no codeword closer to 00101 than 11101, but there *are* codewords equidistant. Indeed 00000 is a codeword at distance 2 from 00101. Statistically, then, we might just as well have decoded 00101 as 00000 — and that is what we *would* have done if we had made a different arbitrary choice of a weight 2 coset leader.
This just goes to show that our error correction is not perfect. It is just the best we can do.
In ball-packing terms, the ball around 00000 of 'radius'

2 intersects the ball around 11101 of radius 2. They intersect in vectors such as 10001 and 00101 and 01100. None of these vectors is any closer to any *other* codeword, so we know, receiving one of these, that at least 2 symbol errors have occurred in transmission.
In trying to correct this we'd like to choose the closest codeword, but there is no unique closest. Thus there is really nothing to choose between guessing 00000 and 11101. We pick arbitrarily from these codewords at distance 2 and hope for the best (or perhaps seek retransmission).

3.142. Exercise. Let C be the 3-ary [4,3]-code generated by

$$G = \begin{pmatrix} 1 & 2 & 2 & 1 \\ 2 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Find a PCM for C. Hence list the codewords of $C^{\perp}$.
Answer:

First we try to get G in standard form. We subtract two lots of row 1 from row 2:

$$G \mapsto \begin{pmatrix} 1 & 2 & 2 & 1 \\ 0 & -3 & -4 & -1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \equiv \begin{pmatrix} 1 & 2 & 2 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Then subtract two lots of row 3 from row 1; then add row 2 to row 3; then swap rows 2 and 3; then mult row 3 by 2:

$$G \mapsto \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 1 & 1 & 1 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 0 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Now viewing this as $G = [1_3|A]$ we put $H = [-A^t|1_{4-3}]$. Thus

$$H = (-2, 0, -1, 1) \equiv (1, 0, 2, 1)$$

This generates $C^{\perp} = \{0000, 1021, 2012\}$.

3.143. Exercise. Let C be the [3,2] code over $F_4 = \{0, 1, a, b\}$ generated by

$$G = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \end{pmatrix}$$

Explain the meaning of the symbols a and b in this field; write down the addition table for it; and hence or otherwise determine the codewords of C and $C^{\perp}$.

3.144. Exercise. Let C be the binary [7,4] code generated by

$$G = \begin{pmatrix} 1000111 \\ 0100110 \\ 0010101 \\ 0001011 \end{pmatrix}$$

(a) (i) find a PCM H for C

(a) (ii) compute $G.H^t$

(b) show that $d(C) = 3$

(c) (i) show that C is perfect

(c) (ii) how many coset leaders have weight 1?

(d) construct a syndrome look-up table for C

(e) decode the received vector 1110100.

3.145. Exercise. Write down a PCM for the binary Hamming [15,11] code.

Answer:

We need to write down all non-zero vectors in $\mathbb{Z}_2^4$ as columns:

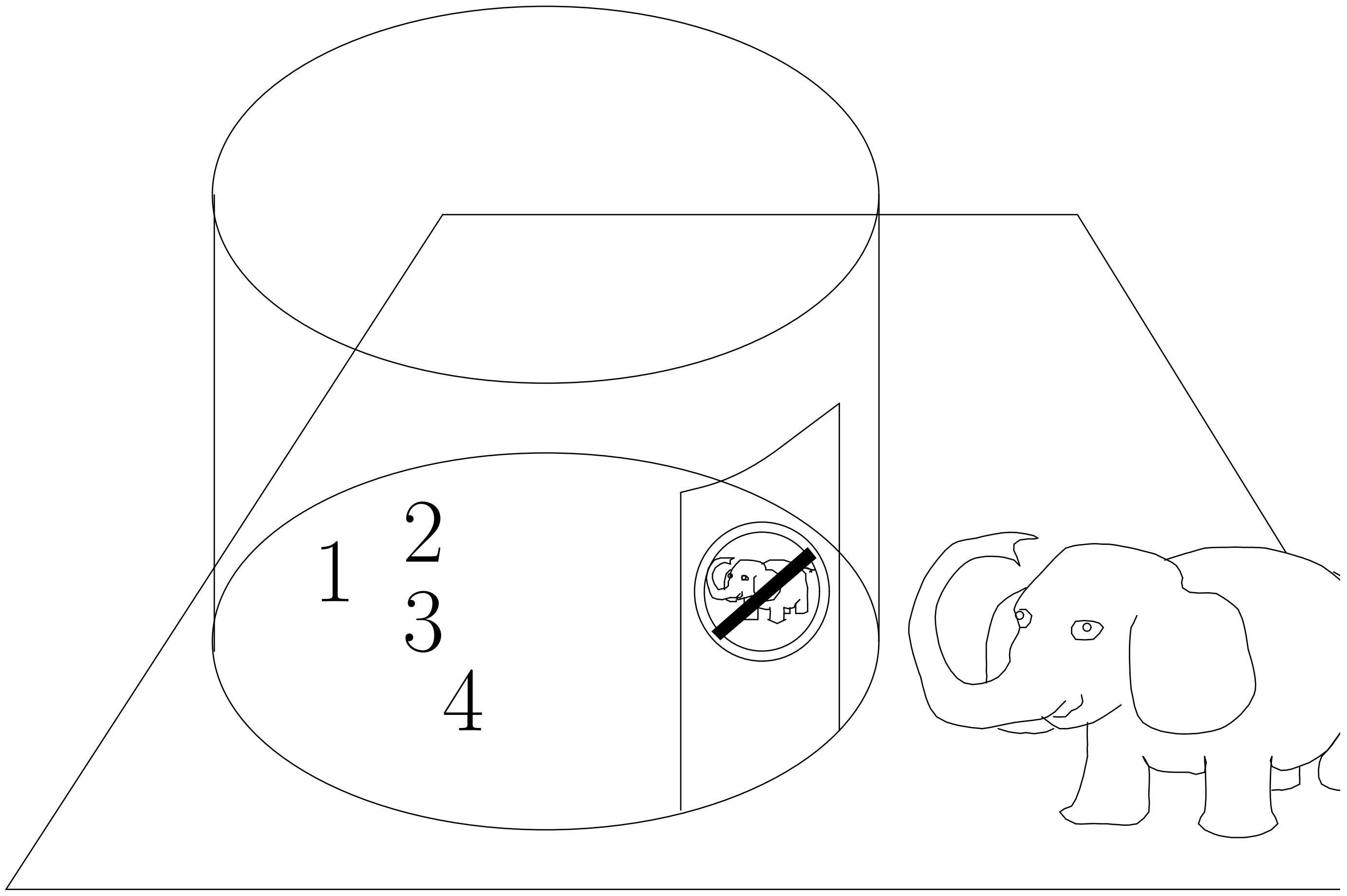
$$H = \begin{pmatrix} 000000011111111 \\ 000111100001111 \\ 011001100110011 \\ 101010101010101 \end{pmatrix}$$

Chapter 4

Mathematical Preliminaries

4.1 Sets

- A SET is a collection of objects.
- A specific set is 'defined' by any means which unambiguously tells us how to determine whether an arbitrary object is in the set or not.
- The objects in a set are called the ELEMENTS of the set. We write $x \in S$ in case x is an element of set S. We write $x \notin S$ otherwise.



Example 1 *Suppose we write* $S = \{1, 2, 3, 4\}$. *Then* S *is a set,* $1 \in S$, *and* $2 \in S$, *but* $5 \notin S$, *and of course* $[\mathfrak{M}y \text{ pet } \mathfrak{E}lephanT] \notin S$ *(see figure).*

Example 2 *Suppose* $T = \{x \text{ an integer} \mid x^2 < 27\}$. *This is another set (here* $|$ *means 'such that'). We have* $-1 \in T$, $6 \notin T$, $-6 \notin T$, $0 \in T$, *and so on.*

Example 3 *Suppose* $V = \{0, 1, 2, 3, 4, 5, -1, -2, -3, -4, -5\}$. *This is also a set. The order in which we write the elements in a set is not important.*

4.1. Definition.[SUBSET] A set S is a subset of a set T if and only if every element of S is an element of T.

A *mathematical notation* representation of this definition is:

$$S \subseteq T \iff (x \in S \Rightarrow x \in T).$$

Make sure you understand what each symbol means, and how to read this line as a sentence in ordinary language! For example, $\subseteq$ is the symbol for subset; $\iff$, also written *iff*, means 'if and only if'; and $\Rightarrow$ is the symbol for 'implies'. The brackets here are a guide to the eye, containing a statement within the sentence which is a composite of other statements.

Example 4 *Comparing S from example 1 and T from example 2 we see that $S \subseteq T$.*

Example 5 *Let $\mathcal{S}$ be the set of playing cards in a 52 card deck of playing cards. Then the set of all 'club' cards is a subset $\mathcal{S}_\clubsuit$ of $\mathcal{S}$. Suppose that a card dealer deals out the pack into 4 equal hands (i.e. 13 cards each). Each of these hands is a subset of $\mathcal{S}$.* What is the probability that one of these hands is $\mathcal{S}_\clubsuit$?

A set $S \subseteq T$ is called a PROPER subset of T (written $S \subset T$) provided at least one element of T is not in S.

We write $S = T$ if $S \subseteq T$ and $T \subseteq S$.

Example 6 *Comparing T from example 2 and V from example 3 we see that $T = V$.*

Exercise 8 *Write down five sets — call them S_1, S_2, S_3, S_4, S_5, say — with the property that $S_i \subset S_{i+1}$ for $i = 1, 2, 3, 4$ (i.e. $S_1 \subset S_2$, and so on).*

Exercise 9 *Show that for A, B, C sets, if $A \subset B$ and $B \subset C$ then $A \subset C$.*

The general procedure for solving this kind of problem is as follows:

State what is to be done in mathematical notation; *if* the solution is very long (not the case here, as we will see!) give a one sentence overview of your plan of attack; convert the known information into mathematical notation (expanding up all terms to their full definitions) and rearrange to achieve the required result....

Solution 9.1 *We need to show that $x \in A$ implies $x \in C$, and that there is some $y \in C$ such that $y \notin A$. Suppose that $A \subset B$ and $B \subset C$. Since $A \subset B$ then $x \in A$ implies $x \in B$. Further since $B \subset C$ then $x \in B$ implies $x \in C$. Altogether then $x \in A$ implies $x \in C$, which* *shows that $A \subseteq C$. But $A \subset B$ also implies that there exists $y \in B$ such that $y \notin A$, and since $y \in B$ implies $y \in C$ then $A \subset C$. QED.*

4.1.1 Sets built from other sets

In what follows, S, T are two sets:

4.2. Definition.[INTERSECTION] We define a new set, the 'intersection of S and T', written $S \cap T$, by

$$S \cap T = \{x | x \in S \text{ and } x \in T\}.$$

For example, if $S = \{1, 2, 4\}, T = \{1, 3, 4, 5, 6\}$, then $S \cap T = \{1, 4\}$.

The EMPTY set, denoted $\emptyset$, is the set containing no objects. For example,

$$\{1, 3, 5\} \cap \{2, 4, 6\} = \emptyset.$$

4.3. Definition.[DISJOINT] We say that two sets A, B are disjoint in case $A \cap B = \emptyset$.

Example 7 *Let W be the set of all those ancient Egyptian pyramids under whose northernmost foundation stone is hidden evidence that aliens once visited the Earth. It is true to say that there is a set E such that $W \supseteq E$,* *since $W \supseteq \emptyset$ and $W \supseteq W$. But is it true that there is a set E such that $W \supset E$?*

4.4. Definition.[UNION] We define a new set

$$S \cup T = \{x | x \in S \ \text{ or } \ x \in T\}.$$

N.B. The 'or' here is the *inclusive or.*
For example, if $S = \{1, 2, 4\}, T = \{1, 3, 4, 5, 6\}$, then $S \cup T = \{1, 2, 3, 4, 5, 6\}$.

Exercise 10 *Verify that $S \cup (T \cup V) = (S \cup T) \cup V$ for all sets S, T, V.*

From this exercise we see that we may speak unambiguously of the union of several sets (i.e. not just two sets).

4.5. Definition.[POWER SET] The power set of a set S, denoted $\mathcal{P}(S)$, is the set of all subsets of S.

Example 8 $\mathcal{P}(\{\emptyset\}) = \{\emptyset, \{\emptyset\}\}$. *Notice how careful one must be with the brackets for this to make any sense.*

In SET THEORY it is useful to have a notion of 'all possible objects' which might be collected together to form sets. Unfortunately this notion is really too vague as it stands. In practice we define a UNIVERSAL set U to be a set containing all possible objects *under discussion* (with the kind of object under discussion being determined, perhaps implicitly, by the context). We usually specify a universal set *for a given problem* as some set which, at least, contains as subsets all the sets in which we are currently interested.

The COMPLEMENT of a set S (with respect to some such universal set U) is written S', and means the set of all objects in U NOT in S.

4.1.2 Cartesian product

4.6. Definition.[CARTESIAN PRODUCT] The Cartesian product of two nonempty sets S and T, written $S \times T$, is the set $S \times T$ given by

$$S \times T = \{(a, b)|a \in S \text{ and } b \in T\}$$

where $(a, b) \in S \times T$ is a constructed object made from the ordered pairing of a and b.

For example,

$$\{1, 2, 3\}\times\{x, y\} = \{(1, x), (1, y), (2, x), (2, y), (3, x), (3, y)\}.$$

Note that the order in which we write the pair $(1, x)$ (say) is important. This pair is a single element of the Cartesian product. The pair $(x, 1)$ is NOT an element of the Cartesian product in our example (but it would be an element of $\{x, y\} \times \{1, 2, 3\}$, so obviously $S \times T \neq T \times S$ in general!).

Example 9 *Let* $H = \{A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K\}$ *— the set of values on the cards in a 52 card deck of playing cards. Then the set* $\mathcal{S}$ *from example 5 may be written*

$$\mathcal{S} = H \times \{\clubsuit, \heartsuit, \spadesuit, \diamondsuit\}$$

where $(2, \clubsuit)$ *represents the two of clubs, and so on. In this notation we might write* $\mathcal{S}_\clubsuit = H \times \{\clubsuit\}$ *(it might be safer to write* $\cong$ *instead of* $=$*, see later). We may similarly introduce* $\mathcal{S}_\heartsuit = H \times \{\heartsuit\}$*, and so on. Note that* $\mathcal{S}_\heartsuit \cap \mathcal{S}_\clubsuit = \emptyset$*; and* $\mathcal{S}_\heartsuit \cup \mathcal{S}_\clubsuit \cup \mathcal{S}_\spadesuit \cup \mathcal{S}_\diamondsuit = \mathcal{S}$*.*

4.1.3 Aside on the subsets of the set of real numbers

We will discuss the topic of real numbers later, but in order to introduce some notation we here note that the set of real numbers, denoted $\mathbb{R}$, has a sequence of subsets:

$$\emptyset \subset \mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}.$$

Exercise 11 *Explain the meaning of each of these symbols.*

The set of NATURAL NUMBERS, denoted $\mathbb{N}$, satisfies *Peano's axioms*:

(a) $1 \in \mathbb{N}$;

(b) for each $n \in \mathbb{N}$ there exists a unique $n' \in \mathbb{N}$ called 'the successor of n', $(n+1)$;

(c) 1 is not the successor of any $n \in \mathbb{N}$;

(d) if $n' = m'$ then $n = m$;

(e) if $S \subseteq \mathbb{N}$ and $1 \in S$ and if $n \in S$ implies $(n+1) \in S$, then $S = \mathbb{N}$.

Let us see what we get using these axioms:

$$\mathbb{N} = \{1, (1+1), ((1+1)+1), (((1+1)+1)+1), ((((1+1)+1)+1)+1),$$

Of course we have a shorthand for this:

$$\mathbb{N} = \{1, 2, 3, 4, 5, ...\}.$$

Exercise 12 *Give an example of a set $\overline{\mathbb{N}}$ which satisfies all of Peano's axiom axiom (c): 1 is not the successor of any $n \in \overline{\mathbb{N}}$.*

Solution 12.1 *Without this axiom we could allow, for example,*

$$(((((1+1)+1)+1)+1)+1) = 1.$$

Then

$$\overline{\mathbb{N}} = \{1, 2, 3, 4, 5\}$$

(and + doesn't have its usual meaning!).

We have lots more to say about sets and numbers. We will come back to them later.

4.2 Relations and Functions

Let S and T be nonempty sets. A RELATION from S to T is any subset of $S \times T$.

For example, if S is the set of Mathematicians, and T is the set of Statisticians, then we might define a relation ρ by writing

$$\rho = \{(a, b) \in S \times T | a \text{ is older than } b\}.$$

It is often convenient to write $a\rho b$ (and say 'a has the relation ρ with b', or 'a stands in relation ρ to b', or in this case simply 'a is older than b') in case $(a, b) \in \rho$.

Note in particular than in this example (and in general) $a\rho b$ does not imply $b\rho a$!

Suppose we have a relation $\rho \subseteq S \times T$. Then

4.7. Definition.[DOMAIN] The domain of ρ, written *dom* ρ, is the set of elements of S which appear as the left hand sides of pairs which are elements of ρ.

For example, if $\rho = \{(1, x), (2, x)\}$ then *dom* $\rho = \{1, 2\}$.

4.8. Definition.[RANGE] The range of ρ, written *ran* ρ, is the set of elements of T which appear as the right hand sides of pairs which are elements of ρ.

In our example, *ran* $\rho = \{x\}$.

4.9. Definition.[INVERSE] The inverse of ρ, written ρ^{-1}, is the set obtained by reversing the order of each pair in ρ.

In our example $\rho^{-1} = \{(x,1),(x,2)\}$.

Let ρ be a relation from S to T. Then it is also a relation from *dom* ρ to T.

Exercise 13 *Show that ρ is also a relation from S to ran ρ.*

Solution 13.1 *This is an example of a simple kind of 'proof' of a claim, where we simply have to insert the definitions of the terms and rearrange a little:*

We have to show that $(a,b) \in \rho$ implies $b \in$ ran ρ. But the definition of ran ρ says that it is the set of all *right hand sides of such pairs, so certainly it includes this one!*

Exercise 14 (compulsory) *By similar means:*

1) Show that a relation ρ is also a relation from dom ρ to any Q such that $Q \supset$ ran ρ.

2) Show that a relation ρ is NOT a relation from dom ρ to any P such that $P \subset$ ran ρ.

4.10. Definition.[FUNCTION] A function is a relation in which each element of the domain appears exactly once as the left hand side of a pair.

Thus $\{(1, x), (2, x)\}$ is a function, but $\{(x, 1), (x, 2)\}$ is not. To generate some more examples let us consider $A = \{a, b, c, d\}$, $B = \{r, s, t, u, v\}$. Then:
(i) $\{(a, t), (c, r), (d, s), (c, v)\}$ is not a function from A to B because c appears twice;
(ii) $\{(a, u), (b, r), (c, s), (d, u)\}$ is a function;
(iii) $\{(a, c), (a, u), (b, s), (c, r), (d, t)\}$ is *not* a function;
(iv) $\{(a, u), (b, u), (c, u), (d, u)\}$ is a function;
(v) $\{(a, r), (b, s), (c, t), (d, u)\}$ is a function.

Recall that we can think of the set of real numbers $\mathbb{R}$ as the set of points on the x-axis of a Cartesian x, y frame. Then the set $\mathbb{R} \times \mathbb{R} = \mathbb{R}^2$ may be represented by the points on the whole plane (i.e. with "coordinates" $(x, y) \in \mathbb{R}^2$). It follows that any subset of the points of the plane is a relation! In particular any line drawn on the plane gives a relation. We are familiar with the representation of functions from $\mathbb{R}$ to $\mathbb{R}$ by this means. On the other hand, we know that only certain lines drawn on the plane correspond to a function (an arbitrary scribble, while giving a perfectly good relation, would not normally be a function). You should compare your intuitive understanding of this with the definition above!

Since each element of the domain appears exactly once in a function, we have the opportunity for a new and neater notation. The right hand side of each pair in the function is uniquely given by the left hand side. We can recognise this by writing the pair $(x, f(x)) \in f$, say. Of course we then often go on to specify the right hand side ”as a function of” the left hand side explicitly, arriving at the more familiar notation for functions, for example:

$$f(x) = 1 + x^2.$$

Altogether we give the concrete definition of a specific function as follows. First we specify the name of the function, and the domain, and the CODOMAIN (which is any set containing the range - this definition may seem rather arbitrary at first, but it is quite convenient, as we will soon see). For example if f is the function, A is the domain and $B \supseteq \ ran\ f$ we write

$$f : A \to B$$

and say ”the function f maps the set A into the set B”. Then we write

$$f : x \mapsto f(x)$$

which means that the action of f on a specific element $x \in A$ is to take it to $f(x) \in B$. In practice at this point we may be able to give $f(x)$ explicitly. For example we might write, altogether,

$$f : \mathbb{R} \to \mathbb{R}$$

$$f : x \mapsto x^3 + 3x - 2.$$

This may seem like a lot of fuss over nothing! It isn't, and you should take care to study it very carefully. There will be examples shortly. First, here are some refinements:

4.11. Definition.[ONTO] A function $f : A \to B$ is called onto (or SURJECTIVE) if $ran\ f = B$.

Note that examples (ii),(iv) and (v) above are NOT onto.

4.12. Definition.[ONE-TO-ONE] A function is one-to-one (or INJECTIVE) if

$$(\ (a, b) \in f \ \text{ and } \ (a', b) \in f\ ) \ \text{ implies } \ a = a'.$$

That is, distinct elements in A have distinct "images" in B ($f(a) \in B$ is called the "image of a under f"). Note that examples (ii) and (iv) above are not one-to-one, but that example (v) is one-to-one.

A function which is not one-to-one is called MANY-TO-ONE.

4.13. Definition.[BIJECTION] A function which is both one-to-one *and* onto is called a bijection.

Exercise 15 *Give three examples of bijections.*

There are various useful pictorial representations of functions. These will be discussed in class.

4.14. Definition.[IDENTITY FUNCTION] For each set A there is a function from A to A, called the identity function, denoted 1_A, and given by

$$1_A : A \to A$$

$$1_A : a \mapsto a.$$

Two functions h and g are said to be EQUAL (written $h = g$) if they have the same domain and codomain, and $h(x) = g(x)$ for all x in the domain. For example, if h, g are two functions from $\mathbb{R}$ to $\mathbb{R}$ given by $h(x) = x+x$ and $g(x) = 2x$, then $h = g$.

The *restriction* of a function f to a subset of the domain is the function on that subset obtained by applying f to it.

4.2.1 Composition of functions

Let $f : A \to B$ and $C \supseteq ran\ f$ and $g : C \to D$. Then

4.15. Definition.[COMPOSITE FUNCTION] The composite function $g \circ f$ is defined by

$$g \circ f : A \to D$$

$$g \circ f : a \mapsto g(f(a)).$$

We write $(g \circ f)(a) = g(f(a))$.

For a relation ρ we understand $\rho(a)$ to be the *set* of objects b such that $a\rho b$. For S a subset of the domain of ρ we understand $\rho(S)$ to be the union of sets $\rho(s)$ over every $s \in S$. Relations are then composable in much the same way as functions.

Although every relation has an inverse (and hence every function has an inverse *as a relation*), not every function has an inverse which is itself a function.

Exercise 16 *Show that the inverse of a function is a function if and only if the function is a bijection.*

Exercise 17 *For $f : A \to B$ a bijection, show that*

$$f \circ f^{-1} = 1_B$$

and

$$f^{-1} \circ f = 1_A.$$

4.2.2 Permutations

If the number of elements in a set is a natural number (i.e. if it is finite, since then it is certainly a non-negative whole number!) then the set is called a finite set. For example, $A = \{a, b, c, d, e, f, g\}$ is a finite set, as it has 7 elements; meanwhile $\mathbb{R}$ is not a finite set. We will return to this point later.

4.16. Definition.[ORDER] The order (or degree) of a finite set A, denoted $|A|$, is the number of elements in the set.

Denoting the set of all finite sets by F, then the 'order' operation is a function

$$\text{Order} : F \to \mathbb{N}$$

i.e.

$$\text{Order} : A \mapsto |A|.$$

For example, if $B = \{a, b, c, d\}$ then $\text{Order}(B) = |B| = 4$.

4.17. Definition.[PERMUTATION] A bijection $f : A \to A$ on a finite set A is called a permutation of A.

For example, if $S = \{1, 2, 3, 4\}$ then f given by $f(1) = 2$, $f(2) = 3$, $f(3) = 4$, $f(4) = 1$ is a permutation. Permutations may be written in the form

$$\begin{pmatrix} a & b & c & ... & x \\ f(a) & f(b) & f(c) & ... & f(x) \end{pmatrix}.$$

This one then becomes

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}.$$

Exercise 18 *Verify that any $f : A \to A$ is 1–to–1 if and only if it is onto.*

Let A be a finite set of degree n, and f be a permutation of A. If repeated composition of f with itself produces the identity function after $|A|$ compositions and not before (or, equivalently, if for any $x \in A$ we have that $\{x, f(x), (f \circ f)(x), (f \circ (f \circ f))(x), ...\} = A$) then the permutation is called a CYCLE of A, or n–CYCLE. More generally, if f restricts to a cycle of B for some $B \subset A$, and acts as $f(x) = x$ for all $x \notin B$, then f is a $|B|$–cycle. (Note that there are no permutations of A which are p-cycles with $p > n$.)

Exercise 19 *Show that the f defined in the example above is a cycle. Give an example of a bijection $g : S \to S$ which is not a cycle.*

Exercise 20 *Let A be the set of letters in the alphabet, together with the hyphen symbol – (so $|A| = 27$). Let A' be the subset of these symbols occuring at least once in the word* gerbil–brain*. Write down A'.*

Let $f : A' \to \mathbb{N}$ be the alphabetical ordering of these symbols (so $f(a) = 1$, $f(b) = 2$, $f(e) = 3$, and so on) with $f(-) = 9$. Note that $|ran\ f| = 9$. What is the word obtained by applying the inverse of f to the sequence of elements of ran f given by 1653791643281?

4.3 Equivalence Relations

Let ρ be a relation from set A to A (i.e. $\rho \subseteq A \times A$). Then

4.18. Definition.[REFLEXIVE/SYMMETRIC/TRANSITIVE]
.

1. ρ is reflexive if and only if $a\rho a$ for all $a \in A$.

2. ρ is symmetric if and only if whenever $a\rho b$ then $b\rho a$.

3. ρ is transitive if and only if whenever ($a\rho b$ and $b\rho c$) then $a\rho c$.

Examples:

ρ = "belongs to the same family as" is reflexive, symmetric and transitive;

$\rho =$ "is an ancestor of" is transitive;

$\rho =$ "is the mother of" is none of these!

4.19. Definition.[EQUIVALENCE RELATION] A reflexive, symmetric, transitive relation is an equivalence relation.

Such a relation is often written $\sim$ (as in $a \sim b$) unless it already has a name.

For specific relations, we usually define a pair, consising of the set A together with its equivalence relation: $(A, \sim)$. Thus we have:

(1) $(\mathbb{N}, =)$ given by $a\rho b$ if and only if $a = b$;

(2) $(\mathbb{Z}, \sim)$ given by $a \sim b$ if and only if $5|(a-b)$ (here we have introduced the following

4.20. Definition.[DIVIDES] For $n, m \in \mathbb{Z}$ we say p divides m, and write $p|m$ (*not* to be confused with p/m), in case the equation $m = pn$ is solved by some $n \in \mathbb{Z}$.

for example here $11 \sim 1$ (i.e. $5|(11-1)$) since $11-1 =$ 5.2 - see later).

Let's check these:

In (1) we have $a = a$ for any number a, so the relation is reflexive; if $a = b$ then certainly $b = a$, so it is symmetric; and if $a = b$ and $b = c$ then $a = c$, so transitive;

(2) is more of a challenge, we have $(a-a) = 0$ and $5|0$, so reflexive; we have $(a-b) = -(b-a)$, so if $5|(a-b)$ then $5|(b-a)$, so symmetric; and finally if $(a-b) = 5k$ (say) and $(b-c) = 5l$ (with $k, l \in \mathbb{Z}$) then $(a-c) = 5(k-l)$, so transitive!
The relation (2) is sometimes written $a \equiv b$ (mod.5).

4.3.1 Equivalence classes

4.21. Definition.[EQUIVALENCE CLASS] Given a pair $(A, \sim)$ we define the equivalence class containing $a \in A$ to be the set

$$[a] = \{x \in A \ : \ x \sim a\}.$$

Note that $[a] \subseteq A$; $a \in [a]$; and if $b, c \in [a]$ then $a \sim b, c \sim a$ and indeed $b \sim c$ (i.e. *any* two elements of the same class are equivalent).
Theorem 4.22.*[On equivalence classes] Let $\sim$ be an equivalence relation on a set A and let $[a]$ be the equivalence class of $a \in A$. Then for any $a, b \in A$*

(i) $[a] = [b]$ if and only if $a \sim b$;

(ii) if $[a] \neq [b]$ then $[a] \cap [b] = \emptyset$.

Proof: The theorem may be broken into three parts.
Firstly, the 'if' part of (i):

We can write this part $[a] = [b] \Leftarrow a \sim b$, so this is what we need to show. In other words we must show that if we *assume* $a \sim b$, then $[a] = [b]$ follows, so.... Let $a \sim b$. Then by definition $a \in [b]$. Then again, $[a] \subseteq [b]$, since if $x \in [a]$ then $x \sim a$, but $a \sim b$ and so by transitivity $x \sim b$, that is $x \in [b]$. Similarly $[b] \subseteq [a]$, so finally $[a] = [b]$.

Now the 'only if' part of (i) (i.e. to show $[a] = [b] \Rightarrow a \sim b$):

If $[a] = [b]$ then since $b \sim b$ we have $b \in [a]$ and so $b \sim a$;

Lastly, part (ii):

We will prove this by CONTRADICTION. This means we assume the *opposite* to what is required, and prove this must be false (if the opposite is false, then logically the statement itself must be true). The trick here is to figure out what the opposite of the statement is! This is not always obvious, but in our case the opposite would be:

a and b can be found such that $[a] \neq [b]$ but $[a] \cap [b] \neq \emptyset$.

Let's assume *this* statement true, and see what happens. Consider such an a and b, and consider any $x \in [a] \cap [b]$. If it exists (the last ingredient of the statement says it does!) then this means $x \sim a$ and $x \sim b$. This then implies $a \sim b$ by symmetry and transitivity. But part (i), which is already proved, says that *this* can only happen when $[a] = [b]$ — a contradiction between the consequences of the first and second ingredients of the statement. The only resolution is that there can be no such x — that is, $[a] \cap [b] = \emptyset$. QED.

There will be more examples of this kind of proof shortly.

4.23. Definition.[PARTITION] Given a set A, if there exists a set I and a collection of nonempty subsets $\{X_i \mid i \in I\}$ of A such that

(i) $x \in A$ implies $x \in X_i$ for some $i \in I$;

(ii) $X_i \cap X_j = \emptyset$ unless $i = j$,

then the collection $\{X_i \mid i \in I\}$ is said to form a partition of A.

So BY THEOREM 1 an equivalence relation $\sim$ on a set A defines a partition of A into its equivalence classes.

Example 10 $(\mathbb{Z}, \sim)$ *where* $a \sim b$ *iff* $(a - b)$ *divisible by 5.*

We have

$$[0] = \{...., -10, -5, 0, 5, 10, 15, ...\}$$

$$[1] = \{...., -9, -4, 1, 6, 11, 16, ...\}$$

$$[2] = \{...., -8, -3, 2, 7, 12, 17, ...\}$$

and $[3], [4]$ *similarly (exercise). Altogether there are five classes partitioning the integers* $\mathbb{Z}$. *Sometimes we write these classes simply as* $0, 1, 2, 3, 4$ *'modulo 5' (or mod 5). Note that* $[0] = [5] = [10] = ...$, *and* $[3] = [8] = [13] = ...$ *and so on.*

The SET OF EQUIVALENCE CLASSES here has five elements and is written $\mathbb{Z}_5$ - sometimes called the set of 'residues' mod 5.

We can do 'mod 5' arithmetic, as in

$$4 + 3 = 2 \qquad\qquad mod\ 5.$$

This makes sense in as much as the sum of any element of $[4]$ with any element of $[3]$ is always some element of $[2]$. (For example $9 + 8 = 17$. Now check it in the general case!). Multiplication also works on residues, in a similar way (check it!).

This can be done for residue classes modulo any integer. For example we have a complete arithmetic 'mod

3':

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

.

Special things happen when the integer is *prime* as here. See later.

Conversely, given a partition of A we can define an equivalence relation on A by $a \sim b$ iff a, b belong to the same set X_i of the partition. For example: Let $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$ with partition $\{1, 2, 0\}, \{3\}, \{4, 5, 7\}, \{6, 8\}, \{9\}$; then the corresponding equivalence relation is

$$\{(1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (7,7), (8,8), (9,9), (0,0), (1,2), (2,$$

$$(0,1), (2,0), (0,2), (6,8), (8,6), (4,5), (5,4), (4,7), (7,4), (5,7), (7,5)\}.$$

4.4 Countability

Consider the collection A of all sets. (We could say the set of all sets. This is a potentially dangerous notion — see [Cohn] on *Russell's Paradox* — but the dangers need not concern us here.) For X, Y sets let $X \sim Y$ iff there exists a bijection $f : X \to Y$. Note

(i) $1_X : X \to X$, so $\sim$ is reflexive;

(ii) If $f : X \to Y$ is a bijection, then $f^{-1} : Y \to X$ is a bijection, so $\sim$ is symmetric;

(iii) $f : X \to Y$ and $g : Y \to Z$ bijections implies $(g \circ f) : X \to Z$ is a bijection, so $\sim$ is transitive.

Altogether then, we have an equivalence relation.

In some sense (and precisely for the finite sets) each equivalence class contains all the sets with the same 'number of elements'.

For a set A the equivalence class $[A]$ is written $Card\ A$ ('Cardinal A'). $Card\ A = Card\ B$ iff A, B are 'numerically equivalent'.

For *finite* sets the equivalence class of all sets containing n elements is sometimes written simply as n. This n is called a 'cardinal number' (in general such numbers are the numbers associated with set sizes - but the finite cardinal numbers are the natural numbers $n \in \mathbb{N}$).

A set in $Card\ \mathbb{N}$ is called COUNTABLY INFINITE. A *countable set* is either finite or infinite. A set is countable, it 'can be counted', if when one sets out to count the elements (i.e. assign a distinct number to each of them from 1,2,3,...) there is a way of doing this such that any given element of the set eventually gets counted. N.B. This is not the same as saying that *all* the elements will be counted in finite time. $Card\ \mathbb{N}$ is sometimes denoted $\aleph_0$ ('aleph zero').

Example:

$$E = \{2, 4, 6, 8, 10, 12, ...\} \subset \mathbb{N}$$

what is the cardinality of E? Well,

$$f : \mathbb{N} \to E$$

$$f : x \mapsto 2x$$

is a bijection (check it!), so $Card\ E = \aleph_0$.

Obviously $\aleph_0$ is not any finite cardinal number. In a sense it is *bigger*. In a similar sense, as we will see shortly, there are still bigger infinite numbers (i.e. there are infinite sets too big to have a bijection with $\mathbb{N}$)! We call $\aleph_0$ a TRANSFINITE NUMBER. We have the following transfinite arithmetic:

$$2\,\aleph_0 \quad = \quad \aleph_0$$

(since naively we threw away half the elements of $\mathbb{N}$ to get E, and yet it didn't change the cardinality)

$$\aleph_0 + \aleph_0 = \aleph_0$$

$$\aleph_0 + 1 = \aleph_0$$

....so, is every infinite set countable? Well, what sets do we know which are bigger than $\mathbb{N}$? Obviously we have the rational numbers - even the set $\mathbb{Q}_+$ of positive rational numbers obeys $\mathbb{Q}_+ \supset \mathbb{N}$, but in fact:

4.24. Proposition. *Card* $\mathbb{Q}_+ = \aleph_0$

Proof: We will list the elements of $\mathbb{Q}_+$ in such a way that a bijection with $\mathbb{N}$ (i.e. a way of 'counting' the elements such that any given element is eventually counted) can be explicitly given.

We organise the elements of $\mathbb{Q}_+$ as follows:

$$\begin{array}{cccccc}
1/1 & 1/2 & 1/3 & 1/4 & 1/5 & ... \\
\\
2/1 & 2/2 & 2/3 & 2/4 & 2/5 & ... \\
\\
3/1 & 3/2 & 3/3 & 3/4 & 3/5 & ... \\
\\
\\
... & & & & &
\end{array}$$

(here many elements are counted more than once, but at least we can be sure that eventually any given element does appear on the grid). Now suppose we consider an arbitrary element, which is of the form $x = p/q$ by definition. Each South-East diagonal of the grid gives all the numbers with fixed $p+q$. We will count through the grid starting from the top left and then counting up each such diagonal in turn (i.e. running through the diagonals in order $p + q = 2, 3, 4, 5, \ldots$). That is, our bijection will be:

$$f(1/1) = 1$$

$$f(2/1) = 2$$

$$f(1/2) = 3$$

$$f(3/1) = 4$$

(2/2 has already been counted as 1/1)

$$f(1/3) = 5$$

$$f(4/1) = 6$$

and so on. QED.

You should check that you *understand* how the one-to-one and onto conditions are satisfied here.

4.25. Corollary. *[Exercise] Card* $\mathbb{Q} = \aleph_0$.

4.4.1 Uncountability

So we still havn't found any bigger 'numbers' than $\aleph_0$, even though $\mathbb{Q}$ contains $\mathbb{N}$ and much much more. What about the even bigger set $\mathbb{R}$?

4.26. Proposition. *Card* $\mathbb{R} \neq \aleph_0$.

Proof: We will prove the proposition by contradiction! In other words we will assume that $\mathbb{R}$ is countable, and prove that this must be wrong.

If we assume that $\mathbb{R}$ is countable then any subset must also be countable (if every element can be counted, then every element of a subset can be counted). Let us consider the set $(0,1) \subset \mathbb{R}$, which is the set of real numbers between 0 and 1. Our assumption implies that $(0,1)$ is countable, so that each $x \in (0,1)$ may be numbered distinctly by some function f, a bijection onto the natural numbers. Since it is a bijection it has an inverse f^{-1}, i.e. for each natural number n there is a unique real number $f^{-1}(n)$ in the interval $(0,1)$.

Now consider an $x \in (0,1)$ written in decimal form. This form may be familiar to you. For example $x = .7 = .70000...$ (recurring 0s) or $x = .76582342222....$ (recurring 2s), or $x = \pi - 3 = .1415926...$ (no recuring pattern!). Note that to avoid duplicating values x we can avoid recurring 9s. To see why recurring nines are redundant consider, say, .79999... (recurring 9s) and .80000... (recurring 0s). The calculation $9 \times .79999... = (10 \times .79999...) - .79999... = 7.9999... - .79999... = 7.2$ shows that $.79999... = .8$. Now consider a particular decimal

$$y = .a_1 a_2 a_3 a_4....$$

(e.g. $y = .2343479....$, so each $a_i \in \{0,1,2,3,4,5,6,7,8,9\}$). Suppose that for all i, the i^{th} decimal place - a_i - is chosen to be *different from* the i^{th} decimal place of the real number $f^{-1}(i)$. For example a_1 is different from the first decimal place of $f^{-1}(1)$; a_2 is different from the first decimal place of $f^{-1}(2)$; and so on.

By this construction y differs from each and every element of the list of images of f^{-1} in at least one decimal place. But of course if two numbers are the same then (excluding the situation with recurring 9) they must be the same in every decimal place, so y is actually a different number from each and every element of the list. But it is of the form $y = .a_1a_2....$, so certainly $y \in (0,1)$. But then f^{-1} is not onto, so it is not a bijection, so neither is f, which is then a CONTRADICTION of the original assumption.

We conclude that the assumption must be wrong, that is $(0,1)$ is 'uncountable'. Then $\mathbb{R}$ is uncountable. QED.

Now we can introduce a new 'number': $Card\ \mathbb{R}$, which is often written C for 'continuum'.

This raises some intriguing questions. For example: Are there any cardinal numbers 'between' $\aleph_0$ and C? Are there numbers bigger than C? The mathematician CANTOR has thought a lot about these problems, with limited success. For the first question we have 'Cantor's continuum hypothesis', in which he claims that there are no cardinal numbers between $\aleph_0$ and C. What do you think?......

For the second question - let us recall the notion of power set $\mathcal{P}(S)$ - the set of all subsets of S.

Exercise 21 *Verify that for finite sets*

$$|\mathcal{P}(S)| = 2^{|S|}.$$

In general considering $Card\ \mathcal{P}(S)$ (which we may abuse notation to write as $2^{Card\ S}$) is a reasonable way of trying to generate new cardinals. Cantor proved that $Card\ \mathcal{P}(S) > Card\ S$ continues to hold for transfinite numbers, so there exist infinitely many transfinite cardinals:

$$\aleph_0, 2^{\aleph_0}, 2^{2^{\aleph_0}}, ...$$

Exercise 22 (Difficult) *Prove that* $Card\ \mathcal{P}(\mathbb{N}) = 2^{\aleph_0} = C$.

4.27. Definition.[$\aleph_1$] We define $\aleph_1$ to be the next bigger cardinal after $\aleph_0$.
This raises the question: Is $\aleph_1 = C$? What do you think?...

4.5 Orderings

Let P be a non-empty set.

4.28. Definition.[Partial Order Relation] A partial order relation on P, usually written $\leq$, is a relation with the following properties:

(i) $x \leq x$ for all $x \in P$ (reflexivity);

(ii) $x \leq y$ and $y \leq x$ implies $x = y$ ('anti-symmetry');

(iii) $x \leq y$ and $y \leq z$ implies $x \leq z$ (transitivity).

Then the pair $(P, \leq)$ is called a partially ordered set, or just a poset for short.

Examples:

(1) $(\mathbb{N}, \leq)$ where $\leq$ means 'less than or equal to' is a poset.

(2) $(\mathbb{N}, <)$ is NOT a poset (it fails reflexivity test).

(3) $(\mathbb{Z}, a$ divides $b)$ is NOT a poset (1 divides -1, and -1 divides 1).

(4) $(\mathbb{N}, a$ divides $b)$ is a poset.

(5) For X any set then $(\mathcal{P}(X), \subseteq)$ is a poset.

Let us check this one:

Reflexivity: $A \subseteq A$ for any set A, so OK.

Anti-symmetry: $A \subseteq B, B \subseteq A$ implies $A = B$, again for any two sets.

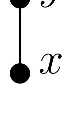
Transitivity: $A \subseteq B, B \subseteq C$ implies $A \subseteq C$, so OK.

(6) For X a nonempty set, the set of all real valued functions $f : X \to \mathbb{R}$, with relation $f \leq g$ iff $f(x) \leq g(x)$ for all $x \in X$, is a poset.

4.5.1 Diagrammatic representation of posets: Hasse diagrams

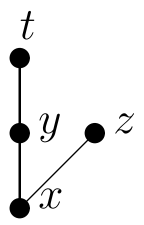
In a Hasse diagram we take advantage of the definition to represent a poset as follows. We draw a 'node' or spot for each element of the set, and a bond between nodes x and y (say) if either $x \leq y$ or $y \leq x$ (if there is some z such that $x \leq z \leq y$ we only draw bonds between x and z and between z and y, since this automatically creates a connection for us between x and y). We draw y ABOVE x on the page if $y \geq x$.

For example: (1) $S = \{x, y\}$ with $x \leq y$ then the diagram is



where $x \leq x$ and $y \leq y$ are to be understood implicitly.

(2) $S = \{x, y, z, t\}$ with $x \leq y, y \leq t, x \leq z$ (and $x \leq x, y \leq y, z \leq z, t \leq t$ and $x \leq t$ implicitly) is



We will give some more examples in the lecture.

4.29. Definition.[Comparability] In a poset $(P, \leq)$ two elements $x, y \in P$ are said to be comparable iff $x \leq y$ or $y \leq x$ (i.e. if joined in the Hasse diagram).

For example, in $(\mathcal{P}(\{1,2,3\}), \subseteq)$ the elements $\{1\}$ and $\{2\}$ are NOT comparable, but $\{1\}$ and $\{1,2\}$ are comparable.
Then again in $(\mathbb{N}, m$ divides $n)$ we have that 4 and 6 are not comparable but 3 and 6 are comparable.

4.30. Definition.[Total Ordering] A poset in which every pair of elements is comparable is called a total ordering, or a linear ordering, or a CHAIN.

For example $(\mathbb{N}, \leq)$ is a chain; $(\mathcal{P}(X), \subseteq)$ is not a chain; and $(\mathbb{N}, a$ divides $b)$ is not a chain.

A linear ordering $\leq$ on a set P in which every non-empty subset has a LEAST ELEMENT (i.e. an element l such that $l \leq x$ for all x in the subset) makes $(P, \leq)$ a WELL ORDERED SET.

4.31. Proposition.*[Exercise]*

1. $(\mathbb{N}, \leq)$ *is well ordered.*

2. Every finite chain is well ordered.

3. $(\{x \in \mathbb{Q} \,:\, x \geq 0\}, \leq)$ *is NOT well ordered.*

4.32. Definition.[BOUNDEDNESS] A poset $(P, \leq)$ in which there exists an element, $\perp$ (say), such that $\perp \leq x$ for all $x \in P$, and an element, $\top$ (say), such that $x \leq \top$ for all $x \in P$ is said to be BOUNDED.

For example:

1. $(\mathcal{P}(S), \subseteq)$ is bounded (even when S is infinite). Exercise: What are $\perp$ and $\top$ here?

2. $(\{1,2,3,4,6,9\}, a$ divides $b)$ has no $\top$, so is not bounded (exercise: draw the diagram).

4.33. Definition.[MAXIMAL/MINIMAL ELEMENTS] In a poset $(P, \leq)$ an element $x \in P$ is MAXIMAL iff $y \geq x$ implies $y = x$ (i.e. x is not $\leq$ any *other* element).

Similarly for MINIMAL elements.

e.g.1. in $(\{1,2,3,4,6,9\}, a$ divides $b)$ the elements 4,6,9 are maximal.

e.g.2. in a bounded poset $\top$ (also called 'the top element') is the unique maximal element, and $\perp$ (also called 'the bottom element') is uniquely minimal.

4.34. Definition.[LOWER BOUND] In $(P, \leq)$ let A be a nonempty subset of P. Then $x \in P$ is a LOWER

BOUND of A if $x \leq a$ for all $a \in A$.

4.35. Definition.[GREATEST LOWER BOUND / INFIMUM] With A as above, x is a GLB (or 'inf') of A if $x \geq$ every lower bound of A.

Exercise 23 *If inf A exists it is unique. Prove it!*

Similarly, y is an UPPER BOUND of A if $y \geq a$ for all $a \in A$, and y is a LEAST UPPER BOUND (or SUPREMUM, or 'sup') if it is $\leq$ every upper bound of A.

Example: $(\mathbb{N}, \text{is a factor of})$ - let $A = \{4, 6\}$, then 12,24,36,... are all upper bounds for A; 1,2 are lower bounds.

Sup $A = 12$ (Lowest Common Multiple (LCM) of 4 and 6)

Inf $A = 2$ (Highest Common Factor (HCF) of 4 and 6).

4.36. Proposition.*[Zorn's Lemma (see later)] A poset P in which every chain has an upper bound has a maximal element.*

There are some more advanced notes on posets (specifically on LATTICES) to be found in the version of these notes published on the maths web pages. Of course, looking at these additional notes is optional.

4.6 Sets with Binary Operations

A (closed) binary operation $*$ on a non-empty set S is a function

$$* : S \times S \to S.$$

If $*((x, y)) = z$ we usually write $x * y = z$.

Note that $x * y$ is defined for all $x, y \in S$ and that $x * y = z \in S$.

Examples:

1. $(\mathbb{N}, +)$, i.e. the natural numbers with operation given by addition;

2. for L a lattice both $(L, \vee)$ and $(L, \wedge)$ give binary operations;

3. $(\mathbb{N}, \times)$, ...muliplication of natural numbers is closed;

4. $(\mathbb{N}, -)$ is NOT closed (here $* : \mathbb{N} \times \mathbb{N} \to \mathbb{Z}$, since, for example, $3 - 4 = -1 \notin \mathbb{N}$), and:

5. $(\mathbb{Z}, -)$: $-$ on $\mathbb{Z}$ is a closed binary operation.

4.37. Definition.[ASSOCIATIVITY] An operation $*$ on S is associative iff

$$(x * y) * z = x * (y * z)$$

for all $x, y, z \in S$.

For example addition in $\mathbb{N}$ is associative:

$$(1 + 2) + 5 = 1 + (2 + 5)$$

(note that this *example* does not constitute a *proof* of associativity on its own), but subtraction is not associative:

$$(1-2)-5 \neq 1-(2-5).$$

With assocative operations we usually write just xyz for $x*(y*z)$.

4.38. Definition.[COMMUTATIVITY] A binary operation $*$ is commutative iff

$$(x*y)=(y*x).$$

For example $1+3=3+1$ (and so on), $a \wedge b = b \wedge a$, but $2-3 \neq 3-2$. We DO NOT assume commutativity.

4.39. Definition.[IDENTITY] The pair $(S,*)$ has a (two sided) identity (usually denoted by e) iff there exists $e \in S$ such that for all $x \in S$

$$e*x=x*e=x.$$

4.40. Proposition. *If $*$ is a closed binary operation with identity then the identity is unique.*

Proof: Assume e, f two identities, then $f=e*f=e$ by definition. QED!

Examples:

1. in $(\mathbb{Z},+)$ the identity is $e=0$;

2. in $(\mathcal{P}(X), \cup)$ then $e = \emptyset$;
3. in $(\mathcal{P}(X), \cap)$ then $e =$? (exercise);

4.41. Definition.[IDEMPOTENT] Any $x \in S$ obeying $x * x = x$ is called an idempotent.

4.42. Definition.[INVERSES] Let $(S, *)$ be a set with a closed binary operation and identity e. An element y is an inverse of x iff $x * y = e = y * x$ (so x is also an inverse of y).

Examples: $(\mathbb{Z}, +)$, inverse of 3 is -3;

$(\mathbb{Q}, \times)$, 1 is the identity and the inverse of 3 is $1/3$.

4.43. Proposition. *For a closed associative binary operation $(S, *)$ with identity e, if x has an inverse then the inverse is unique.*

Proof: Let y, z be inverses of x. Then $x * y = e$ and $z * x = e$ so

$$(z * x) * y = e * y = y$$

and

$$z * (x * y) = z * e = z$$

so $y = z$ by associativity. QED.

As a notation, if we write xy for $x * y$ then we write x^{-1} for the inverse of x.

4.6.1 Groups

4.44. Definition.[GROUP] A group $(G, *)$ is a set G with a closed associative binary operation $*$ such that:

1. there exists an (unique) identity;
2. there exists an (unique) x^{-1} for each $x \in G$.

Examples: $(\mathbb{Z}, +)$ is a group;

$(\mathbb{Q}, \times)$ is NOT a group (0 has no inverse); but $(\mathbb{Q} - \{0\}, \times)$ is a group;

$(L, \vee)$ is not a group.

4.45. Definition.[ABELIAN] A group $(G, *)$ with $*$ commutative is called a commutative or abelian group.

Such groups are often written $(G, +)$ even though the operation may not be the usual addition (but in particular $(\mathbb{Z}, +)$ is abelian).

$(\mathbb{Z}_3, +)$ is an abelian group (see the table at the end of section 4), but $(\mathbb{Z}_, \times)$ is not a group (because 0 does not have an inverse, again see the end of section 4);

4.46. Proposition. *Let S_n be the set of permutations of a set A consisting of n objects (recall, from definition 18, that this is the set of bijections $f : A \to A$), and let $\circ$ be the binary operation given by composition of functions, then $(S_n, \circ)$ is a group.*

Proof: We have to check that the operation is closed and associative, that there is an identity, and that each element has an inverse.

Firstly, the composition of two bijections from A to A is a bijection from A to A, so we have closure. Secondly

$$f \circ (g \circ h) : A \to A$$

is given by

$$f \circ (g \circ h)x :\mapsto f(g(h(x)))$$

as is $(f \circ g) \circ h$, so we have associativity. The bijection 1_A acts as the identity, and since the functions are bijections they all have inverses as functions which also serve as their group inverses. QED.

4.47. Proposition.*[Exercise] The group $(S_3, \circ)$ is not abelian.*

4.48. Definition.[SUBGROUP] Let $(G, .)$ be a group. Then $(H, .)$ is a subgroup of G iff $H \subset G$ and $(H, .)$ is a group.

Example: Consider the group $(S_3, \circ)$. The set S_3 consists of elements which we can write (as in section 3.2):

$$1_A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} ; a = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix} ; b = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$$

$$a \circ b = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix} ; b \circ a = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} ; a \circ (b \circ a) = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}.$$

This is the complete list, since the bottom rows should give all possible permutations of the list $1, 2, 3$, of which there are 6 possibilities (i.e. 3!=3.2.1 possibilities). As an exercise you should check, for example, that $b \circ (a \circ b)$ doesn't give anything new (i.e. check closure).

We have labelled the identity element 1_A, as appropriate. The identity element MUST appear as an element of any subgroup (since the subgroup is also a group). In fact the smallest subgroup would be $(\{1_A\}, \circ)$, that is, the group containing *only* the identity element (which is its own inverse).

Another subgroup is $(\{1_A, a\}, \circ)$. You should check closure. Is this subgroup abelian?

We write $H \leq G$ if H, G are groups (we often supress explicit reference to the binary operation for brevity) and H is a subgroup of G. Note that subgroup is a much stronger condition than subset. H a subgroup of G implies H a subset of G, but NOT the other way round.

For example, $\{a, b\}$ is a subset of S_n, but it is not a subgroup because it is not closed, and does not contain the identity (exercise).

Chapter 5

Graphs

To read this section you will need to know about sets. If necessary, see the section on sets, and return here when you are done.

Graph Theory has applications in Sorting and Searching, Combinatorics, queuing theory, programming, cartography, Physics, systems engineering, representation theory, and a host of other areas. It is aslo interesting (to Mathematicians) in its own right.

5.1 Definitions

(**5.1.1**) A *directed graph* (or digraph) is a pair of sets (V, E) together with a pair of functions i and f from E

into V.

The elements of V are the *vertices* of the graph, and those of E are the *edges.* If $i(e) = v_1$ and $f(e) = v_2$ then e is an edge from v_1 to v_2.

Diagrammatically we may represent a digraph as a collection of dots on the page (the vertices), together with a collection of lines (the edges) with direction arrows on them, each starting at some vertex and finishing at some vertex.

A *graph* is similar to a digraph, except that the edges have no direction arrows.

A *simple* (di)graph has no pair of vertices with multiple edges between them.

(**5.1.2**) A *loop* is an edge having the same initial and final point.

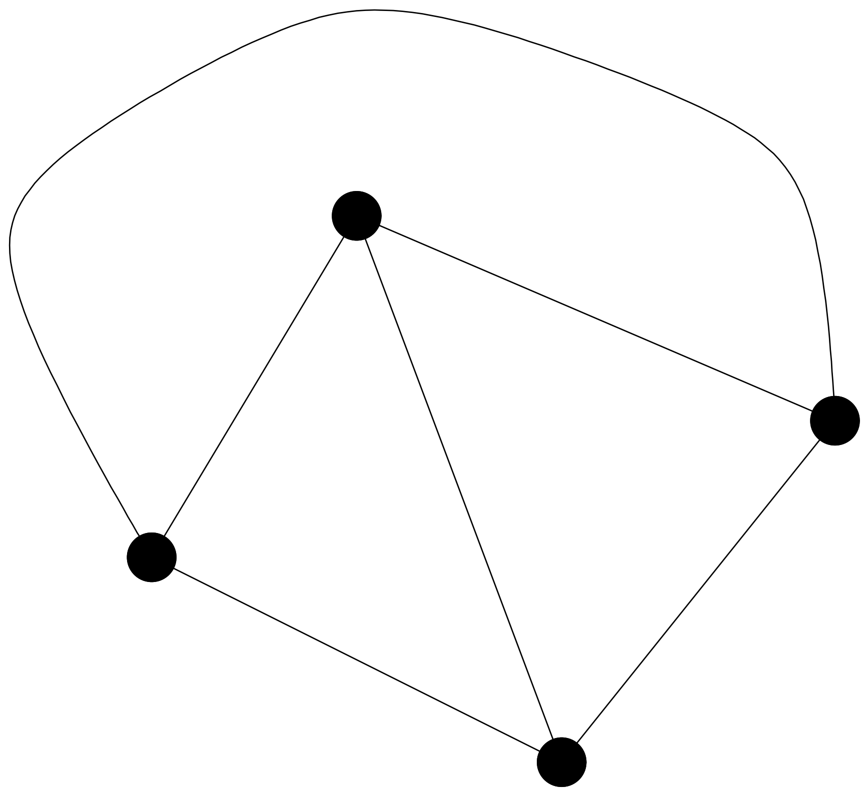
If $G = (V, E)$ and $G' = (V', E')$ are graphs then G' is a subgraph of G if $V' \subseteq V$ and $E' \subseteq E$. A special kind of subgraph is an *induced subgraph.* If G' is an induced sugraph of G then every edge of G for which both endpoints are present in V' is present in E'.

A graph is *connected* if for any two vertices there is a sequence of edges joining one to the other.

A (di)graph is *planar* if it can be represented on the plane in such a way that edges only ever touch at their endpoints (the vertices).

The *degree* of a vertex in a graph is the number of edges incident with that vertex.

The *complete graph* K_n on n vertices is the loop–free graph in which every pair of vertices is connected directly by an edge. For example, K_4 may be represented as:



(**5.1.3**) Execises.

What is the largest n for which K_n is planar?

Discuss the representation of elements of the group S_n as graphs.

(**5.1.4**) A *path* from v_1 to v_2 is a sequence of edges with the first beginning at v_1, each subsequent one beginning where the previous one finished, and with the last one ending at v_2. (On a graph either end of an edge may be regarded as the beginning, whereupon the other end must be *the* end.)

A *circuit* is a path from any vertex x to itself with at least one edge, and no edge occuring twice. For example, a loop is a circuit.

(**5.1.5**) A *tree* is a connected graph without circuits.

(**5.1.6**) Let G be a (di)graph whose vertices are totally ordered. There is a map from the set of such graphs with n vertices to the set of $n \times n$ matrices, as follows. $M_{ij}(G)$ is the number of edges from vertex i to vertex j. (Thus if G is a graph, not a digraph, the matrix is symmetric.)

We say that two (di)graphs, G and G', are *isomorphic* if their vertices can be ordered in such a way that $M(G) = M(G')$.

Note that if, from a given order of the vertices of G we reorder by exchanging two vertices, then $M(G)$ will change by having the corresponding pair of rows interchanged, and the corresponding pair of columns interchanged. Note that this is a similarity transformation (i.e. it can be achieved by conjugating $M(G)$ by an appropriate matrix). Note also that any reordering can be achieved by a sequence of such vertex pair reorderings. (We are back with the symmetric group again!) Hence any reordering acts like a similarity transformation on $M(G)$.

Note also that the order on vertices is not usually deemed to be intrinsic to the graph, so unless an order is specified, any of the versions of $M(G)$ in the same similarity class are equally good representations of G. (Although of course the vertices of G *are distinguishable.*) However, if G and G' are simply different orderings of the same vertex set, let us say by a pair permutation of vertices v and v', and $M(G) = M(G')$, then the graph has a kind of symmetry under swapping v and v'.
For example, if w is an eigenvector of $M(G)$: $wM(G) = \lambda M(G)$, then the entries in w are naturally associated to vertices. If P is the similarity transform matrix which interchanges v and v' then $M(G) = PM(G')P^{-1}$, and with the symmetry: $M(G) = PM(G)P^{-1}$. Thus $wM(G)P = w(PM(G)P^{-1})P = wPM(G) = \lambda wP$, so wP is an eigenvector of $M(G)$ with the same eigenvalue. Note that wP is related to w simply by exchanging the entries corresponding to the vertices v and v'. Then if, for example, the eigenvector associated to the eigenvalue λ

is know to be unique (up to scalar multiplication, as usual), we have that the entries corresponding to the vertices v and v' must be *the same*.
These observations can be useful in computing PageRank-like properties of graphs.

Exercise 24 *What do matrix addition and multiplication correspond to at the level of graphs?*

5.2 Colouring

(**5.2.1**) Let C be any set (for example, but not necessarily, a set of colours). A colouring of a loop–free graph G with C is a function f from V to C such that if there is an edge between any v_1 and v_2 then $f(v_1) \neq f(v_2)$.
(Note that the restriction to loop–free graphs is not really necessary in this definition, but there is *no* colouring of a graph with a loop, since the loop imposes the unsatisfiable condition that the corresponding vertex is coloured differently from *itself*!)

For any loop–free graph its 'chromatic number' is the smallest degree of C such that there exists a colouring of G with C. For example, the chromatic number of K_n is n (exercise!).

Exercise 25 *Let G be a graph and let $C_G(Q)$ denote the number of ways of colouring G with at most Q colours. (For example, $C_{K_2}(Q) = Q(Q-1)$.) Show that $C_G(Q)$ may be expressed as a polynomial in Q for every graph G.*

For graphs G and G' having no vertices in common, let $G \cup G'$ denote the natural disconnected composite graph. For a graph G with distinct vertices v and v' connected by an edge e, let G_e denote the subgraph differing from G only in the absence of edge e, and let $G_{vv'}$ denote the graph obtained from G by identifying v and v' and discarding e.
Note that the transformation $G \mapsto G_e$ (any e) takes a simple graph to a simple graph, but the transformation $G \mapsto G_{vv'}$ does not necessarily do so. For example, picking any e in K_3 we have that $(K_3)_{vv'}$ is a graph like K_2 but with two edges instead of one.

Exercise 26 *Verify the polynomial identity $C_G(Q) = C_{G_e}(Q) - C_{G_{vv'}}(Q)$.*

Hint: note that G and G_e have the same vertex sets. Pick a suitable example and consider which colourings of G_e become 'illegal' when e is re-inserted. How can we count the number of such colourings using $G_{vv'}$?

A more mundane hint is to start by considering some examples.
If $G = K_2$ then $(K_2)_e$ is $K_1 \cup K_1$ (suitably interpretted), and $(K_2)_{vv'} = K_1$, so the claim is that $C_{K_2}(Q) = Q^2 - Q$, which is correct (see above).
Suppose G is like K_2, but with the edge tripled! Then removing one edge does not change the number of possible colourings, i.e. $C_G(Q) = C_{G_e}(Q)$. But identifying the two vertices leaves two edges which are now loops, even after removing the edge e, so $C_{G_{vv'}}(Q) = 0$ and the identity holds.

(**5.2.2**) Example application — Scheduling. Suppose various jobs are to be done in an industrial process producing some product. Suppose that it is possible to perform some of the jobs simultaneously, but not others (perhaps they require the *same* piece of machinery). A schedule for the jobs may be obtained as follows. The set of jobs is represented as vertices in a graph, joining any two if and only if they CANNOT be peformed at the same time. A *colouring* of this graph by a set of *times* then effectively assigns times for performance of jobs in such a way that jobs which cannot be performed simultaneously *are* not scheduled together.

5.3 Planar graphs

(**5.3.1**) Consider the plane with its natural metric topology. We will call a subset of the plane *open* if it is open with respect to this topology. (For example, consider a circle drawn in the plane — the set of all those points interior to, but not actually touching, the circle is an open subset.) An open subset of the plane is *connected* if it cannot be cut into two non–empty open pieces (roughly, it is an open region with the property that we may move continuously from any point within it to any other without leaving the region). Suppose that a graph G has a representation in the plane. A *face* of this representation is a bounded maximal connected region of the plane disjoint from the vertices and edges representing the graph.

An edge is *external* if it has a face on at most one side of it.

5.1. Proposition. *Let finite non–empty connected planar graph G have vertex set V, edge set E, and a planar representation with face set F. Then*

$$|V| + |F| - |E| = 1.$$

Proof: By induction on $|E|$. The base $|E| = 0$ is trivial. Assume the result true for all graphs with $|E| <$

n and let $G = (V, E)$ have n edges (and face set F). Without loss of generality let e be any external edge of G, with ends v_1 and v_2. If the subgraph G' got by removing e is connected then the inductive hypothesis applies to it. It has the same number of vertices, one fewer edge and (since e was external) one fewer face than G. Thus, in this case, the hypothesis holds on G.

If on the other hand removing e separates G into disconnected components then we have two subgraphs (V', E') and (V'', E''), say, with $V' \cap V'' = \emptyset$, $E' \cap E'' = \emptyset$, $V' \cup V'' = V$, $E' \cup E'' \cup \{e\} = E$, and similarly $F' \cap F'' = \emptyset$, and $F' \cup F'' = F$. Both sugraphs must obey the inductive hypothesis, so

$$|V|+|F|-|E| = |V'|+|V''|+|F'|+|F''|-(|E'|+|E''|+1) = 1+1-1 = 1$$

as required. $\square$

(**5.3.2**) The famous 4 colour theorem asserts that any finite planar graph can be coloured with no more than 4 colours. It is clear that there are planar graphs requiring at least 4 colours (see K_4 above). The fact that no more are required is one of the most intriguing and tantalising results in Graph Theory. The known proof is very complicated, although many authors (including the present one) have expended much effort in trying to produce a slicker one. For our purposes it will suffice to prove a weaker result.

5.2. Proposition. *Any finite loop–free planar graph can be coloured with no more than 5 colours.*

Proof: We may assume G to be simple and connected. Proceed by induction on the number of vertices. If there are 5 or fewer vertices then the result follows trivially, so the base is clear. Now assume all graphs with fewer vertices than G may be coloured with 5 or fewer colours. We will show that G must have a vertex of degree ≤ 5 and build a 5 colouring of G around such a vertex, thus establishing the inductive step.

Let M be the number of pairs (e, f) where e is an edge of face f. Each edge bounds at most 2 faces, so

$$M \leq 2|E|$$

and each face contributes at least 3 to M since G is simple, so

$$M \geq 3|F|.$$

Thus

$$|E| \leq |E| + (2|E| - 3|F|) = 3|V| - 3$$

(using Euler's formula from proposition 5.1 at the last).

Let V_i be the number of vertices of G of degree i, then

$$2|E| = \sum_i iV_i$$

and so

$$\sum_i iV_i \leq 6|V| - 6$$

giving

$$\sum_{i=1}^{5}(6-i)V_i \geq 6 + \sum_{i=7}^{k}(i-6)V_i$$

where k is the maximum degree in G. In particular

$$\sum_{i=1}^{5}(6-i)V_i \geq 6$$

and at least one of V_1, V_2, V_3, V_4, V_5 is non-empty!

Now let P be a vertex of degree ≤ 5. Let G' be the induced subgraph of G on $V \setminus \{P\}$. Graph G' is still planar and has fewer vertices, so it can be 5 coloured by hypothesis. Consider such a 5 colouring of G'. We will show how to modify this colouring so that a colouring of the whole of G is possible.

If less than 5 colours are used for the neighbours of P in the colouring of G' then we can colour P with the fifth colour, so assume that P has degree 5 and its neighbours use all 5 colours. (Obviously we have to make a modification so that only 4 are used, while preserving the colouring of G'.) Label the neighbours by number clockwise around P. Suppose there is a path from 1 to 3 using only vertices coloured with two colours (red and blue, say); then there can be no such path from 2 to 4 (any path from 1 to 3 cuts a path from 2 to 4, since the graph is planar). Thus we may assume that at least one of these pairs does not have a two colour path between them. Without loss of generality, then, let 1 and 3 be a pair without a two colour path between them. Now let us say red is the colour of 3, and blue is the colour of 1. To colour the graph G *including* P, colour P red, and change the colour of 3, and any vertex in a red–blue path from 3, to the other one of these two colours.

To see that this works, it remains to check that we still have a colouring of G'. Both 1 and 3 are now blue, but they are not adjacent. All the red–blue paths of changed vertices starting at 3 are still consistent with colouring (red–blue–red becomes blue–red–blue, and so on). Further, there is no vertex adjacent to, but not on, any of these paths which is coloured either red or blue (else it *is* on a path and so changed to be consistent by construction). □

5.4 Exercises

Exercise 27 *You are in a game show on TV. The host shows you three closed doors. Behind one of the doors is a prize. You have to guess which door leads to the prize. There are no clues, so you guess at random, but after you have guessed the host opens one of the other doors and shows that there is* not *a prize behind that door. She then asks if you would like to change your guess, or stay with your original guess.*

Is it better to change, or stay, or does it make no difference to your probability of success?

Draw a graph which illustrates the relevant probabilities and hence gives the answer to this well-known old puzzle.

Exercise 28 *Show that isomorphism is an equivalence relation on the set of all graphs.*

Exercise 29 *Up to isomorphism, how many different simple, loop-free graphs are there with 2 vertices? How many are there with 3 vertices? How many are there with 4 vertices?*

Exercise 30 *Up to isomorphism, how many different simple, loop-free graphs are there with n vertices and*

$\frac{n(n-1)}{2} - 1$ edges? How many are there with n vertices and $\frac{n(n-1)}{2} - 2$ edges?

Exercise 31 *If we have a planar representation of a graph, then this 'restricts' to a planar representation of any subgraph (by simply erasing the relevant vertices and edges). However, it is not always possible to extend a planar representation of a subgraph to a planar representation of a planar graph by adding appropriate objects. Give an example to show this.*

An *Euler tour* of a graph is a walk that uses each edge of the graph once.

Euler's theorem states that if connected graph G has an Euler tour then there are either no vertices of odd degree, or two vertices of odd degree. Conversely, if there are no vertices of odd degree then there is an Euler tour, and it may start at any vertex but must end at the same vertex; while if there are two vertices of odd degree then there is an Euler tour, it must start at one of the vertices of odd degree and end at the other one.

Exercise 32 *(Harder) Prove Euler's theorem.*

A *spanning tree* for a connected graph G is a tree which is a subgraph of G whose vertex set coincides with that of G.

Exercise 33 *Construct an algorithm which takes a graph G as input and produces a spanning tree of G as output.*

Chapter 6

Sorting and Permutation

As we will see, the process of sorting is very much in the realm of pure discrete mathematics. However, even to say what *is* sorting, and why it is worth doing, become complex philosophical questions when viewed in a purely mathematical light. One advantage of coupling with the issue of sorting as a computing problem is that this gives us a collection of applications which serve to put these questions on a more concrete footing. Thus we will proceed, as far as possible, with reference to the mathematical and computational aspects *at the same time.*

The definitive text on sorting is, perhaps, Knuth [**?**].

6.1 Introduction

What is sorting? According to the WordNet online dictionary *to sort* is "to arrange or order by classes or categories" (that is, to put things together which are of the *same sort*). It is convenient to add to this that the classes then be put into some order, and we will usually assume that this means a total order. Indeed, this second part of sorting may also achieve the first since, if we have a hierarchical classification scheme and a total order of all (sub)classes then lexicographic ordering automatically arranges objects into a sequence consisting of runs of objects in the same class (just as a dictionary order groups together all words beginning with A).

Why sort? Arranging objects into classes has many uses. In particular it helps with searching. Depending on the nature of the search either a total order or a classification may be the more useful. For example, if I am looking for a book in a library it is useful if the books are totally ordered. If I am looking for *information* in a library it may be more useful to have all the books on my subject grouped together, for browsing.

Exercise: Let $\{R_i \mid i = 1, 2, 3, ..., n\}$ be a row of books on a shelf. Let us say that the books are labelled each by an integer (representing a book title!). Thus the order of the books on the shelf may be represented as a sequence of integers. Give an algorithm to sort the first 5 of these book into non–descending order of their labels.

An answer: Let $\{I_i \mid i = 1, 2, 3, 4, 5\}$ denote the current list (i.e. possibly after reordering). For $i = 1, ..., 4$ compare I_i, I_{i+1} and swap if $I_i > I_{i+1}$. Iterate this loop until there is no further change.

Example:

$$43521 \to 34521 \to 34251 \to 34215$$

$$\to\to 32415 \to 32145 \to\to 23145 \to 21345 \to\to 12345$$

Q1. Does this process converge in general?
A1. Yes. Note that in each loop the largest numbered book not yet in its correct position is moved to its correct position.
Q2. Is it optimal?
A2. This is the interesting question, but it is not yet well posed, since different algorithms require different operations to be performed. Before we can ask a better question we need some more algorithms for comparison. The above procedure is called “Exchange Sort”.

We could also do:

"Insertion Sort": Construct a new sequence as follows. Locate an empty shelf (hopefully very nearby!). Put R_1 somewhere in the middle to start a new sequence. Then for $i = 2, 3, 4, 5$ insert R_i in the correct position relative to the existing form of the new sequence. Example

$$4\rangle 34\rangle 345\rangle 2345\rangle 12345$$

"Selection Sort": Locate the lowest numbered book and put it on the left on a new shelf. Then locate the lowest numbered remaining book and put it on the new shelf to the immediate right of the previous addition, and iterate.

"Enumeration Sort": For each R_i count the number of R_js which should be to its left. The final list of numbers so obtained gives the order in which to arrange the books on a new shelf. Example:

$$43521$$

$$32410$$

We now have enough algorithms to play with. To decide on optimality we need to know relatively how much effort is required for each of their component operations, and of course how many times (typically) each operation would have to be performed. The first of these data will depend on the system to be sorted (light books/ heavy books etc.). The second can be well addressed in the framework of pure combinatorics. In particular, at the heart of Exchange Sorting is the act of permuting elements of a list. The set of possible acts of permutation form a *group* under composition, and the study of this group can be seen to inform much of the theory of sorting. Accordingly we will start with (and here concentrate mainly on) a study of this group.

We begin by recalling the basic algebraic structures we will use.

6.2 Preliminaries

Before proceeding you will need to have read the section on groups in chapter 4. A good reference book for our purposes might be the Schaum Outline Series volume on *Groups* (but the following is abstracted in part from works of Knuth, Jacobson[**?**], Cohn[**?**, **?**], Maclane and Birkoff[**?**], Bass[**?**], and Green[**?**]).

6.2.1 Algebraic systems

There follows a list of definitions in the form
ALGEBRAIC SYSTEM $A = (A \text{ a set}, n\text{−ary operations})$, axioms.
Extended examples are postponed to the relevant sections.

(**6.2.1**) SEMIGROUP $S = (S, \square)$, $\square$ closed associative binary operation on S.

(**6.2.2**) MONOID $M = (M, \square, u)$, $(M, \square)$ a semigroup, u a unit $(au = a = ua)$.
Example: $(\mathbb{N}_0, +, 0)$.

(**6.2.3**) GROUP $G = (G, ., u)$, G a monoid, $\forall a \in G \exists a'$ such that $aa' = u = a'a$.

(**6.2.4**) ABELIAN GROUP $G = (G, +, 0)$, G a group, $a + b = b + a$.

(**6.2.5**) RING $R = (R, +, ., 1, 0)$, $(R, +, 0)$ abelian group, $(R, ., 1)$ monoid, $a(b + c) = ab + ac$, $(a + b)c = ac + bc$.

(**6.2.6**) INTEGRAL DOMAIN K, K a ring, . commutative, $0 \neq 1$, $mn = 0$ implies either $m = 0$ or $n = 0$.

(**6.2.7**) FIELD F, F integral domain, every $a \neq 0$ has multiplicative inverse.

Exercise 34 *Show that the complex numbers are a field when taken with the usual binary operations of addition and multiplication.*

In fact the complex numbers are almost the only example of a field which we will need in this course. If you don't like the idea of fields, just replace all reference to them with a reference to the system of complex numbers!

Before we forget about general fields altogether though, consider the following exercises.

Exercise 35 *Show that the system of arithmetic mod 3 introduced in Chapter 4 is a field.*

Exercise 36 *Construct the addition and multiplication tables for the system of arithmetic mod 2 (generalising those introduced in Chapter 4). Show that this is a field.*

We will call these systems $\mathbb{F}_3$ and $\mathbb{F}_2$ respectively.

Exercise 37 *Verify that the definition of group above coincides with that given in Chapter 4.*

6.2.2 Categories

(**6.2.8**) A CATEGORY A is a collection of ‘objects’ (the possible failure of this collection to be a set will not concern us here), together with a non–empty set $A(M, N)$ of ‘morphisms’ for each ordered pair (M, N) of objects, and an associative composition

$$A(M, N) \times A(L, M) \to A(L, N)$$

such that there are identities $1_M \in A(M, M)$.

Example: Let **Set** be the collection of all sets, and for $M, N \in \mathbf{Set}$ let $\mathbf{Set}(M, N)$ be the set of maps from M to N. The usual composition of maps is associative and has identities, so this is a category.

Let **Ab** be the collection of all abelian groups and $\mathbf{Ab}(M, N)$ the set of group homomorphisms from M to N. This is a category.

$f \in A(M, N)$ is an ISOMORPHISM if there exists $g \in A(N, M)$ such that $gf = 1_M$ and $fg = 1_N$.

(**6.2.9**) The DUAL CATEGORY A^o has the same objects and composition is reversed $(A(M, N) = A^o(N, M))$.

(**6.2.10**) A (covariant) FUNCTOR $F : A \to B$ is a map on objects together with a map on morphisms which preserves composition and identities.

A CONTRAVARIANT FUNCTOR from A to B is a functor from A^o to B (examples later).

6.3 Groups and representations

For F a field let $\mathfrak{GL}_n(F)$ denote the set of invertible $n \times n$ matrices over F. These matrices form a group under matrix multiplication.

Exercise 38 *Verify this explicitly for $F = \mathbb{C}$ in case $n = 2$.*

Exercise 39 *Write down* all *elements of $\mathfrak{GL}_2(\mathbb{F}_2)$.*

(**6.3.1**) A group homomorphism is a map $\rho : G \to H$ between groups such that $\rho(xy) = \rho(x)\rho(y)$.

Exercise 40 *Construct a group homomorphism between the groups $\mathfrak{GL}_2(\mathbb{C})$ and $\mathfrak{GL}_3(\mathbb{C})$.*

Put $\ker \rho = \{g \in G \mid \rho(g) = e\}$ (where e is the group identity element); and $\operatorname{im} \rho = \{\rho(g) \mid g \in G\}$.

(**6.3.2**) A SUBGROUP S of a group G is a group which is a subset of G. A NORMAL subgroup N of a group G (denoted $N \trianglelefteq G$) is a subgroup which is a union of conjugacy classes of G (thus $n \in N$ implies $gng^{-1} \in N$ for all $g \in G$).

(**6.3.3**) For S any subgroup of G we may partition G into LEFT COSETS OF S as follows. For $g \in G$ the coset

$$Sg = \{sg \mid s \in S\}$$

and the set of left cosets is denoted G/S.

6.1. Proposition. *If $N \trianglelefteq G$ then G/N is a group with multiplication $(Na)(Nb) = N(ab)$, and there is a natural group epimorphism $\pi : G \to G/N$ given by $\pi(g) = Ng$.*

Proof: We need to show that if $a' \in Na$ and $b' \in Nb$ then $N(a'b') = N(ab)$. WLOG put $a' = n_1 a$, $b' = n_2 b$ with $n_i \in N$, then $an_2a^{-1} \in N$ by the definition of normal subgroup, and so

$$N(a'b') = Nn_1an_2b = Nan_2b = Nan_2a^{-1}ab = Nab$$

□

(**6.3.4**) Let $\phi : G \to H$ be a group homomorphism. Then $\operatorname{im}\phi$ is a subgroup of H and $\ker\phi$ is a normal subgroup of G. There is a factorisation $\phi = \phi' \circ \pi$ where $\pi : G \to G/\ker\phi$ is as in proposition 6.1 and $\phi' : G/\ker\phi \to \operatorname{im}\phi$ is the isomorphism $\phi'(\ker\phi g) = \phi(g)$.

(**6.3.5**) A REPRESENTATION of a group G over a field F is a homomorphism $\rho : G \to \mathfrak{GL}_n(F)$ for some n.

A finite group is a group which is a finite set.

(**6.3.6**) A group G ACTS on a set W if there is a map from $G \times W$ to W (denoted $(g, w) \mapsto gw$) such that $1w = w$ and $g(hw) = (gh)w$.

6.4 The symmetric group

(**6.4.1**) A PERMUTATION p of a finite set S is a bijection from S to $\{1, 2, \ldots, |S|\}$, that is to say, an arbitrary total ordering of S. We may represent a permutation by listing the elements of S in the order $p = (p^{-1}(1), p^{-1}(2), \ldots)$.

(**6.4.2**) Let $\mathcal{P}_n$ denote the set of $n \times n$ matrices whose row vectors are a permutation of the standard ordered basis of $\mathbb{C}^n$. These matrices form a group under matrix multiplication, denoted S_n.

The set of bijections of any set T of degree n to itself form a group under composition of bijections, and this group is isomorphic to S_n (the identity bijection maps to the identity matrix). We will confuse the two groups willy–nilly under the name SYMMETRIC GROUP.
It is convenient to use $T = \{1, 2, \ldots, n\}$.

(**6.4.3**) There is a useful pictorial representation of S_n, obtained by tracking the timelines in a sort of some notionally ordered set. We see from figure 6.1 that each act of permutation may be viewed as a collection of trajectories between two rows of vertices (representing the objects in the set before and after rearrangement). Composition is then by juxtaposition of such diagrams.

(**6.4.4**) Abstract to diagrams (i.e. discard the books and keep the trajectories). These form a group isomorphic to S_n. The multiplication of diagrams is to juxtapose them top to bottom (just as they are juxtaposed in the figure, once the books are removed), and then to equate diagrams which realise the same connections from top to bottom.

Exercise: Familiarise yourself with these diagrams and their multiplication. Given the diagram for some $f \in S_n$, what does the diagram for f^{-1} look like?

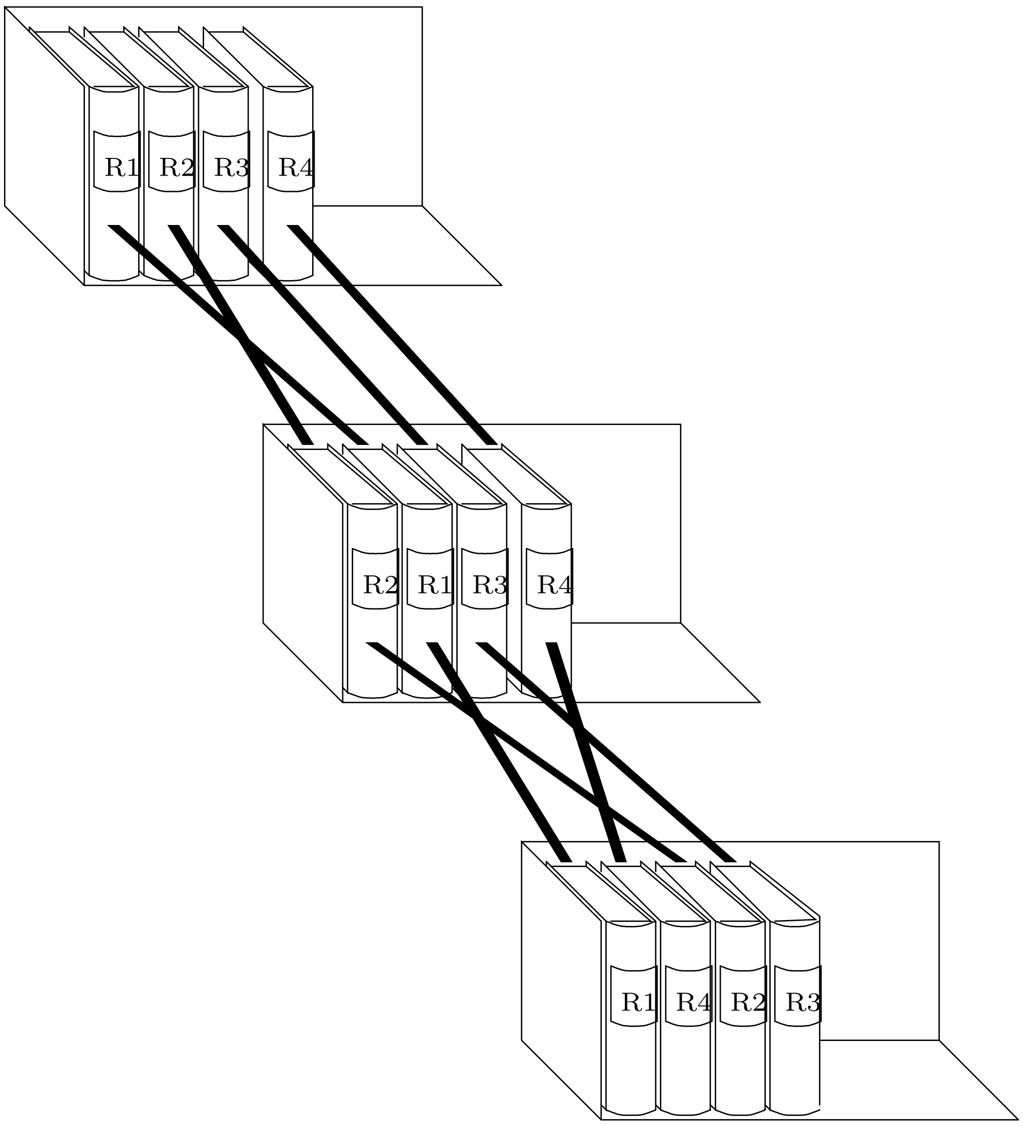


Figure 6.1: An act of permutation as a rearrangement of books on a library shelf. The first step permutes an adjacent pair of books; the second step is a more complicated rearrangement.

(**6.4.5**) In CYCLE notation a permutation p of $T' \subseteq T$ is interpreted as an element f of S_n by

$$\begin{aligned} f(i) &= \quad i \quad && (i \notin T'), \\ f(p_{|T'|}) &= \quad p_1 && \text{and} \\ f(p_i) &= \quad p_{i+1} && \text{otherwise.} \end{aligned}$$

We will call such an element a cycle of length $|T'|$.

There is a SHUFFLE action on the set of cycles of T' obtained by taking

$$p \mapsto (p_2, p_3, \ldots, p_{|T'|}, p_1).$$

The orbits of this action may be represented by the elements in which the numerically lowest element of T appears first. Note that two different cycles may produce the same element of S_n — in particular they do so iff they are in the same shuffle orbit.

Note also that only a subset of elements of S_n are produced this way. However

6.2. Proposition. *A cycle on T' commutes with a cycle on $T'' \subseteq T$ if $T' \cap T'' = \emptyset$.*
Each $f \in S_n$ is a product of such disjoint cycles.

We will write such products so that a longer cycle comes before a shorter one. If we also arrange for the lowest element of each cycle to be written first, and in the case of a tie on length for the cycle with the lowest first element to be written first, then such products are in bijection with S_n. The cycle structure λ of $f \in S_n$ is then the list $(\lambda_1, \lambda_2, \ldots)$ of cycle lengths in the given order (and hence a PARTITION OF n — we write $\lambda \vdash n$, and denote the set of partitions of n by Λ_n).

6.3. Proposition. *Two elements of S_n are in the same conjugacy class iff they have the same cycle structure.*

Exercise: Prove this proposition (hint: draw the diagram for a non–trivial conjugation and so determine the effect of conjugation on a cycle).

6.4.1 Matrix representations

(**6.4.6**) Consider the list of all possible initial orderings of the row of books in figure 6.1. For n distinguishable books there are $n!$ orderings. Let us consider writing out all these orderings as a list (for $n = 3$ we would have 123, 132, 213, 231, 312, 321). A particular act of permutation, such as illustrated in the figure, takes each one of these orderings to a different one. We can encode its effect on the complete list as a matrix as follows.

To begin let us consider the first (upper) act of permutation in the figure (and neglect its effect on the last book, $R4$). In cycle notation the element of the permutation group S_4 in question is (12). Its action takes 1234 to 2134, and in particular *restricts* to take 123 to 213. If the initial ordering had been 1342, say, instead of 1234, the same action would have taken 1342 to 3142 (that is, we have the act of permutation permuting the objects in certain positions, rather than permuting objects with certain labels). Altogether we find that $(123, 132, 213, 231, 312, 321)$ is taken to $(213, 312, 123, 321, 132, 231)$.

We can realize this reordering by matrix multiplication. Specifically, consider the vector space with basis this list of orderings, and the matrix which transforms a 6–tuple whose entries are the basis elements in the original order into one in the new order:

$$(123, 132, 213, 231, 312, 321)\begin{pmatrix} 001000 \\ 000010 \\ 100000 \\ 000001 \\ 010000 \\ 000100 \end{pmatrix} = (213, 312, 123, 321, 132, 231$$

Again ignoring $R4$, the rearrangement whose cycle notation is (23) is captured similarly by

$$(123, 132, 213, 231, 312, 321)\begin{pmatrix} 010000 \\ 100000 \\ 000100 \\ 001000 \\ 000001 \\ 000010 \end{pmatrix} = (132, 123, 231, 213, 321, 312$$

In this way we build a matrix $R(x)$ for each element $x \in S_n$. If we now discard the vectors, we find that the matrices give us a representation of S_n (Exercise: Explain exactly how this works).

(**6.4.7**) Further, if some of the books are duplicates we still get a representation (exercise). We thus have a plethora of representations. Our next job is to bring some order to this situation.

Until further notice we will concern ourselves with complex representations (i.e. where the field of the matrix entries is $\mathbb{C}$).

(**6.4.8**) Suppose M is a representation of a group G. Then conjugating every representation matrix by a fixed (complex) matrix gives another representation isomorphic to the first. For example

$$M(x)M(y) = M(xy) \Rightarrow (A^{-1}M(x)A)(A^{-1}M(y)A) = (A^{-1}M(xy)A).$$

Isomophism is an equivalence relation. The set of representations isomorphic to a given representation is its equivalence class.

(**6.4.9**) Suppose M_1 and M_2 are two representations of G. For each $g \in G$ construct the *matrix direct sum* of $M_1(g)$ and $M_2(g)$. This set of matrices forms a new representation of G.

If a representation is isomorphic to a representation formed in this way then it is said to be REDUCIBLE, otherwise it is IRREDUCIBLE.

(**6.4.10**) Since every representation can be built up simply from irreducible representations, the most important aspect of the study of a group (at least from the point of view of modelling) is the study of its IRREDUCIBLE representations.

(**6.4.11**) A handy indicator is:

6.4. Proposition. *If any matrix A which is not a multiple of the identity matrix obeys $AM(g) = M(g)A$ for every $M(g)$ in a representation then the representation is reducible.*

Exercise: Prove it. Hints:

1. Note that every matrix which is the image of a group element in some representation is necessarily invertible. Thus it has maximal rank, and so a maximal number of eigenvectors. Thus it is diagonalisable.

2. Start by proving that: Two diagonalisable matrices can be simultaneously transformed into diagonal form (i.e. by the *same* transformation) iff they commute.
Exercise: Prove that every irreducible representation of an abelian group is 1–dimensional.

(**6.4.12**) Let g be any element of G. Then the set of elements of G which commute with g (such as e and g) form a subgroup of G (called the normaliser of g).
Let $[g]$ denote the class of g.

6.5. Proposition. *For $g \in G$, $\{gh \mid h \in G\} = G$.*

Proof: Let $f \in G$ be arbitrary. Then $f \in \{gh\}$ since $f = g(g^{-1}f)$. Done.

(**6.4.13**) Consider the K–vector space with basis G. The group multiplication may be extended K–linearly to a multiplication on this space, whereupon it is called the K–group algebra of G, denoted KG. A subalgebra is a subset closed as a space and under this multiplication. A K–representation of G gives rise to a representation of KG (and any subalgebra) by the obvious K–linear extension.

6.6. Proposition. *The number of equivalence classes of irreducible representations of a group G is equal to the number of classes of G.*

We will prove this fundamental result shortly. First we need to do some preparatory work.

(**6.4.14**) Write $G = \{g_i \mid i = 1, 2, \ldots, N\}$ and $|[g]| = N_{[g]}$. Consider the elements $g_j g g_j^{-1} \in [g]$ as g_j runs over G. If g_j is in the normaliser G_g of g then $g_j g g_j^{-1} = g$. Indeed elements $g_j \in G$ in the same coset of G_g give the same conjugate of g; and elements $g_j \in G$ in different cosets of G_g give different conjugates of g. Altogether, then, as we run over all possible choices of g_j in $g_j g g_j^{-1}$, we visit each member of $[g]$ precisely $\frac{N}{N_{[g]}}$ times. Therefore, in the group algebra

$$\sum_{j=1}^{N} g_j g g_j^{-1} = \frac{N}{N_{[g]}} \sum_{g' \in [g]} g'.$$

Indeed, let us use $[g]$ in the group algebra to denote the sum of all the elements in the class, and say that an element of the group algebra is a linear function of classes if it is a linear combination of such basic class sums. We have

6.7. Proposition. *$x \in \mathbb{C}G$ commutes with all $g \in G$ iff x is a linear function of classes.*

Proof: (if) It is enough to show that g commutes with every basic class sum. We have

$$g\left(\sum_{g' \in [h]} g'\right) g^{-1} = \frac{N_{[h]}}{N} g \left(\sum g_j h g_j^{-1}\right) g^{-1} = \frac{N_{[h]}}{N}\left(\sum g g_j h (g g_j)^{-1}\right) =$$

by proposition 6.5.

(only if) If $x = \sum_i k_i [g_i]$ commutes with all $g \in G$ then

$$x = \frac{1}{N}\sum_j g_j x g_j^{-1} = \frac{1}{N}\sum_i k_i \sum_j g_j g_i g_j^{-1}$$

which is a linear function of classes. Done.

(**6.4.15**) *Proof of proposition 6.6* A matrix representation R of G is also a representation of the subalgebra generated by the basic class sums. Since this latter is commutative every element can simultaneously be written in diagonal form. If R is reducible then it is isomorphic to a representation in which the matrices for all elements of G have the same block diagonal structure (exhibiting the irreducible content). In this representation the block components for matrices representing elements of the class sum subalgebra are scalar multiples of the (block) unit matrix (by proposition 6.4). That is, different diagonal terms in these matrices correspond to different irreducible representations.

Since a class function depends on the choice of a free parameter for each class, the representation matrix can depend on at most this many parameters (depending on whether the representation is faithful or not). Thus there can be up to this many different diagonal terms (precisely this many if R is faithful), and thus there are this many inequivalent irreducible representations of G. Done.

(**6.4.16**) For example, there are seven irreducible representations of S_5, corresponding to the seven classes (corresponding in turn to the seven partitions of 5, which are $\{(5),(4,1),(3,2),(3,1,1),(2,2,1),(2,1,1,1),(1,1,1,1,1)\}$).

(**6.4.17**) EXERCISES.

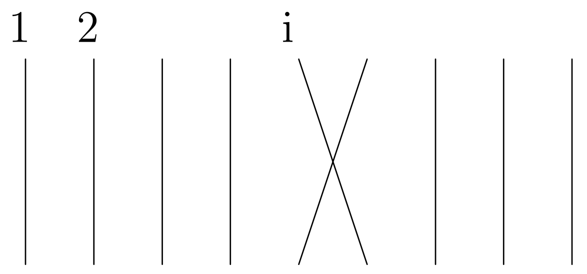
Write down the two irreducible representations of S_2 (corresponding to the partitions of 2, which are $\{(2),(1,1)\}$). Now try to figure out the three irreducible representations of S_3!

Prove proposition 6.2. Hint: ...

6.4.2 Sorting

(**6.4.18**) A group G is GENERATED by a subset S if every element of the group is expressible as a 'product' of one or more of these elements.

(**6.4.19**) Examples. Let us write σ_i for the 'elementary transposition' $(i\; i{+}1) \in S_n$. That is, as a diagram:

$\sigma_i =$ 

We have

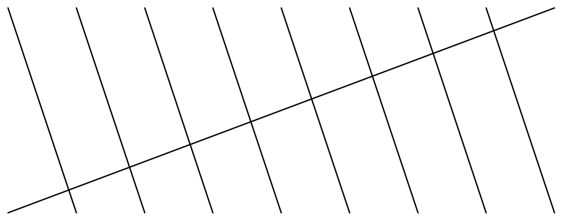
$$\sigma_i \sigma_i = 1 \tag{6.1}$$

$$\sigma_i \sigma_{i\pm1} \sigma_i = \sigma_{i\pm1} \sigma_i \sigma_{i\pm1} \tag{6.2}$$

$$\sigma_i \sigma_j = \sigma_j \sigma_i \qquad\qquad j \neq i \pm 1 \tag{6.3}$$

and $\{\sigma_i \mid i = 1, 2, ..., n-1\}$ generates S_n. (Exercise: Verify this by considering diagrams. More surprisingly, the equations 6.1–6.3 are sufficient for abstract objects $\{\sigma_i \mid i = 1, 2, ..., n-1\}$ to generate a group isomorphic to S_n even if we don't know anything else about them!)

Let $t_n = (12...n)$, i.e. $t_n == \sigma_{n-1}\sigma_{n-2}...\sigma_1 =$

$(12...n) =$ 

Then $\{(12), (12...n)\}$ generates S_n. (To see this note $(t_n)^n = 1$ so $t_n^{-1} = (t_n)^{n-1}$ and

$$t_n^{-1}\sigma_1 t_n = \sigma_2$$

and so on. Thus we reduce to the previous problem.

(**6.4.20**) If $S \subset G$ generates G then the matrices $R(S)$ completely determine a representation R of G.

(**6.4.21**) We can now consider how many elementary transpositions are required to sort an arbitrary permutation. Since each $\sigma_i = \sigma_i^{-1}$ the number required is equal to the minimum number of σ_is required to be multiplied together to achieve that permutation (starting, as it were, from the 'trivial' permutation). It will be evident that the number of σ_is required to build $g \in S_n$ is not unique (e.g. $1 = \sigma_i\sigma_i = \sigma_i\sigma_i\sigma_j\sigma_j$ etc.). However, there is a well defined minimum number required, which we will call len(g) (pronounced 'length g') — we normally say len(1) = 0; then len((23)) = 1 and so on.

The length of g is equal to the number of crossings of lines in the diagram of g, provided all the lines are drawn straight (and the rows of vertices are slightly agitated randomly so that no more than two lines ever intersect at the same point). Alternatively, the minimum number can be read off from the permutation in a way directly analogous to the "bubble" sort of the permutation (a kind of exchange sort — see Knuth).

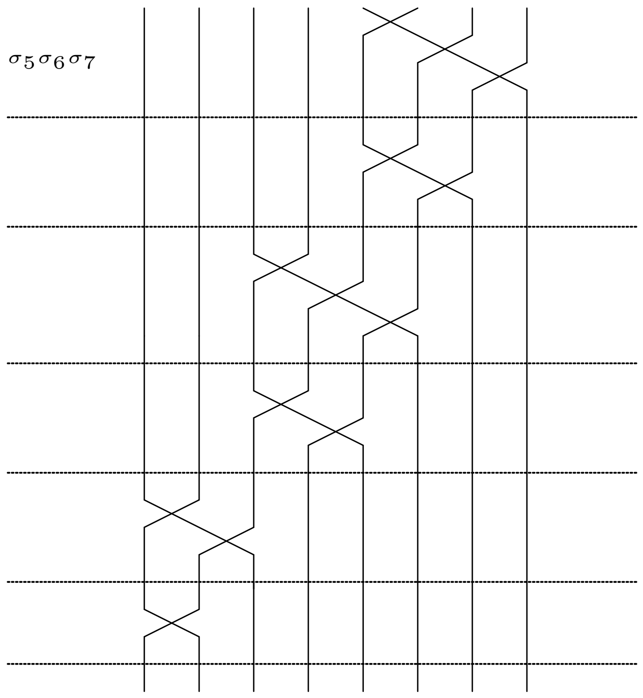


Figure 6.2: An act of permutation as a product of elementary transpositions, organised following the 'bubble' sorting algorithm.

For an example let us consider the permutation

$$\begin{pmatrix} 12345678 \\ 32658714 \end{pmatrix} = (458)(1367).$$

How can this be expressed minimally as a product of elementary transpositions? One (non–unique) way is illustrated in figure 6.2. We first locate the object which should be permed into 'last place', and move it into last place by a sequence of elementary transpositions. Then repeat for the object which should be in next to last place, and so on. (nominal bubble sort pictures some of the elementary transpositions occuring in parallel, if they commute, but this does not change the number of elementary transpositions required). Note that this construction ensures that no two lines cross each other more than once, and hence, provided the required 'perm' is achieved (!), that the number of elementary transpositions is minimal.

We can see from this that the maximum number of factors which could be required is

$$(n-1)+(n-2)+\ldots+1=\frac{n(n-1)}{2}$$

and that this is only achieved by, for example,

$$\begin{pmatrix} 12345678 \\ 87654321 \end{pmatrix} =: \ w_0.$$

6.4.3 Exercises

1. Determine the mean and a median for the length of elements of S_n. That is, determine the average number of elementary transpositions required to sort an arbitrary permutation of n objects into a specific total order. (Hint: If you are stuck you should try to determine the lengths of all elements of S_n for $n = 1, 2, 3, 4$ and then consider how to generalise your findings.)

2. (Non–compulsory, for 10 bonus points) Determine the modal length(s) of elements of S_n.

3. Determine all one–dimensional representations of S_n. Give a non–trivial relationship between one of these representations and the len function.

4. Determine if the following gives a representation of S_3

$$\sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$\sigma_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

and if so, determine if that representation is irreducible.

5. Show that the set L of all elements of the form $(i\ j)$ $(i < j,\ i, j \in \{1, 2, ..., n\})$ generates S_n. Describe an algorithm to sort a permutation of n objects using this set. Discuss the practical differences between using this set to sort books (in the library at Alexandria) and using the set of elementary transpositions. Define a length function on S_n appropriate for the set L, and relate this to the normal length function.

Chapter 7

Hardware

This component of the course is lab based. We will attempt to separate and identify within an arbitrary PC the following components: Motherboard; Processor (CPU); BIOS chip; RAM; Hard disk; Floppy disk; IO ports; Network capability; Video subsystem; HCI devices (keyboard, mouse etc.). We will discuss the role of these components and their relationship with the operating system and other software. We will attempt to reassemble the system and, if functional (!), to install an operating system. If *this* can be done we will attempt to create a LAN (local area network) with another such PC, by installing appropriate Network Interface Cards, cables, and hubs, and configuring these devices in software (both at the driver level and at the Internet Protocol level).

Exercise 41 *Make sure you can physically identify the core modular components (as listed above) of a generic PC.*

Exercise 42 *Draw a graph encoding the modular structure of a typical PC.*

Exercise 43 *Draw a graph encoding the structure of a typical small LAN.*

Exercise 44 *Under what circumstances might a single PC contain two NICs?*