Interestingly, *no finite projective plane of order n is known* if n is not a power of a prime; it is in fact conjectured that in that case no finite projective planes exist. For small values of n, this means that finite projective planes of order n exist for $n = 2, 3, 4, 5, 7, 8, 9, 11$. So the first question arises when $n = 6$. In this case, we can prove that the answer is negative as follows: Assume there exists a finite projective plane of order six. That plane would have 43 vertices. The Bruck–Ryser theorem, given in Exercise 8, shows that, in that case, the equation

$$x^2 + z^2 = 6y^2 \tag{8.4}$$

would have a solution in which x, y, and z are all integers and not all equal to zero. Let us assume that such a solution exists, and let (x, y, z) be a solution in which $|x| + |y| + |z|$ is minimal. Note that, in any solution, each of x and z has to be divisible by three. Indeed, otherwise their square would be of the form $3k + 1$, and therefore the left-hand side would not be divisible by three, which is a contradiction. However, as both x and z are divisible by three, the left-hand side is divisible by nine, and so y must also be divisible by three. This implies that $(x/3, y/3, z/3)$ is also an integer solution to (8.4) and that $|x/3| + |y/3| + |z/3| < |x| + |y| + |z|$, which is a contradiction.

Therefore, no finite projective plane of order six exists. The next value of n to discuss is $n = 10$. This is a very difficult case, but the answer is still known to be negative. See Supplementary Exercises 9 and 10 for further negative results on the existence of finite projective planes.

Quick Check

1. Prove that the Fano plane cannot be drawn in the plane so that each edge corresponds to a straight line segment.
2. Let $n \geq 3$, and let S be a set of n points in the plane so that there is no straight line that contains all points in S. Is it true that there are two points $x, y \in S$ so that the straight line spanned by x and y does not contain any other points in S?
3. Let G be a complete graph on six vertices. Consider the design whose vertices are the 15 edges of G and whose blocks are the 15 perfect matchings and 20 triangles in G. Prove that this design is balanced and regular.

8.3 Error-correcting codes

8.3.1 Words far apart

We could probably tell apart a bar and a dentist's office, even if both lacked a few of the usual features. These places are so different that it suffices to see

parts of them to tell which one is the bar and which one is the dentist's office. This idea turns out to be of crucial importance in Coding Theory.

Let us assume we want to send a message from our cell phone using just the two-letter binary alphabet consisting of the letters 0 and 1. Say the message that we want to send is a YES or NO message. We could agree with the recipient that 1 means yes, and 0 means no. This is simple enough if we are both sure that we will not make any mistakes in typing.

However, if mistakes are possible, then this way of encoding messages will not be efficient. Indeed, one single mistake could totally turn the meaning of the message into its opposite. One way to make sure that our message is not misunderstood is to send it over and over again, in consecutive bits. Say that we will send our message three times. If the message is YES, then we will send the digits 111, and if the message is NO, then we will send the digits 000. These two codewords are not at all similar to each other. Therefore, if we are sure that at most one typing mistake will be made, we can rest assured that our message will be understood properly. Indeed, if we want to send the codeword 111 (resp. 000), and at most one mistake will be made, then the received word will contain at least two 1s (resp. at least two 0s). So as long as at most one bit is erroneous in each codeword, *all errors can be corrected*.

This simple example can be generalized in many different directions. First, it could be that there are more than just two possible messages to send. Second, it could also be that there are more than two digits in our coding alphabet. Third, more than one mistake may be made during typing. Nevertheless, the main idea of our simple example is crucial. This idea is that *if the codewords are sufficiently dissimilar* from each other, then we can tell them apart *even if a few mistakes are made*.

It is time that we made the notions of "sufficiently dissimilar" and "few mistakes" more precise.

Definition 8.18 *Let x and y be words of the same length over the same finite alphabet. Then the* Hamming distance *of x and y, denoted by $d(x, y)$, is the number of positions in which x and y differ.*

Example 8.19 *If $x = 100101$ and $y = 001100$, then $d(x, y) = 3$, since x and y differ in the first, third, and sixth positions.*

Now that we have a notion of distance between two words, we can define the concept of *spheres* and *balls* as well. These will be defined in a way analogous to Euclidean geometry.

Definition 8.20 *Let x be a word of length k over a finite alphabet A. Then the sphere $S_r(x)$ of center x and radius r is the set of k-letter words over A that are of distance r from x.*

Similarly, the ball $B_r(x)$ of center x and radius r is the set of k-letter words over A that are of distance at most *r from x.*

Example 8.21 *Let A be the binary alphabet, let $k = 4$, and let $x = 1010$. Then*

$$S_1(x) = \{0010, 1110, 1000, 1011\},$$

while

$$B_1(x) = \{1010, 0010, 1110, 1000, 1011\}.$$

It is now easy to express our previous observations in a more precise way.

Proposition 8.22 *Let C be a code over a finite alphabet that consists of codewords of the same length. Let us assume that, for every pair of codewords x and y of C, the equality $B_r(x) \cap B_r(y) = \emptyset$ holds. Then, as long as at most r digits are erroneous in each codeword, all errors can be corrected.*

If a code C has the property that all errors can be corrected if at most r mistakes per codeword are made, then we will call that code *r-error-correcting*.

Fine, the reader could say, but how can we quickly check that the balls $B_r(x)$ and $B_r(y)$ are indeed disjoint for every pair (x, y) of codewords? The reader has surely learned the *triangle inequality* in high school. This inequality said that, in Euclidean geometry, the sum of two sides of a triangle is always larger than its third side, essentially because the shortest path between two points is by a straight line. Fortunately, something very similar is true for words over a finite alphabet.

Lemma 8.23 (Triangle inequality for words) *Let x, y, and z be words of the same length over a finite alphabet. Then*

$$d(x, y) \leq d(x, z) + d(z, y).$$

A proof is given in Exercise 10.

Corollary 8.24 *Let C be a code consisting of words of the same length over the same finite alphabet. Let us assume that, for every pair (x, y) of codewords in C, the inequality $d(x, y) \geq 2r + 1$ holds. Then C is r-error-correcting.*

Proof: It suffices to prove that, for each pair (x, y) of words in C, the equality $B_r(x) \cap B_r(y) = \emptyset$ holds, and our claim will follow from Proposition 8.22.

Now let us assume that $z \in B_r(x) \cap B_r(y)$. Then the triangle inequality implies that

$$d(x, y) \leq d(x, z) + d(z, y) \leq r + r = 2r,$$

which is a contradiction. So such z does not exist. $\Diamond$

Very good, you might say again, but I still have to check that no two codewords are closer than $2r + 1$ to each other. How can I do that quickly? This would not be a problem if space were not a concern. Indeed, it is very