CHAPTER 1

Introduction

This book is about counting. Of course this doesn't mean just counting a single finite set. Usually, we have a family of finite sets indexed by a natural number n, and we want to find $F(n)$, the cardinality of the nth set in the family. For example, we might want to count the subsets or permutations of a set of size n, lattice paths of length n, words of length n in the alphabet $\{0,1\}$ with no two consecutive 1s, and so on.

1.1 What is counting?

There are several kinds of answer to this question:

- An explicit formula (which may be more or less complicated, and in particular may involve a number of summations). In general, we regard a simple formula as preferable; replacing a formula with two summations by one with only one is usually a good thing.
- A recurrence relation expressing $F(n)$ in terms of n and the values of $F(m)$ for $m < n$.This allows us to compute $F(0), F(1), \ldots$ in turn, up to any desired value.
- A closed form for a *generating function* for F. We will have much more to say about generating functions later on. Roughly speaking, a generating function represents a sequence of numbers by a power series, which in some cases converges to an analytic function in some domain in the complex plane. An explicit formula for the generating function for a sequence of numbers is regarded as almost as good as a formula for the numbers themselves.

If a generating function converges, it is possible to find the coefficients by analytic methods (differentiation or contour integration).

In the examples below, we use two forms of generating function for a sequence $(a_0, a_1, a_2, \ldots)$ of natural numbers: the *ordinary generating function*, given by

$$\sum_{n\geq 0} a_n x^n,$$

and the *exponential generating function*, given by

$$\sum_{n\geq 0} \frac{a_n x^n}{n!}.$$

We will study these further in the next chapter, and meet them many times during later chapters. In Chapter 10, we will see a sort of explanation of why some problems need one kind of generating function and some need the other.

- An asymptotic estimate for $F(n)$ is a function $G(n)$, typically expressed in terms of the standard functions of analysis, such that $F(n) - G(n)$ is of smaller order of magnitude than $G(n)$. (If $G(n)$ does not vanish, we can write this as $F(n)/G(n) \to 1$ as $n \to \infty$.) We write $F(n) \sim G(n)$ if this holds. This might be accompanied by an asymptotic estimate for $F(n) - G(n)$, and so on; we obtain an *asymptotic series* for F. (The basics of asymptotic analysis are described further in the next section of this chapter.)
- Related to counting combinatorial objects is the question of generating them. The first thing we might ask for is a system of sequential generation, where we can produce an ordered list of the objects. Again there are two possibilities.

  If the number of objects is $F(n)$, then we can in principle arrange the objects in a list, numbered $0, 1, \ldots, F(n) - 1$; we might ask for a construction which, given i with $0 \leq i \leq F(n) - 1$, produces the ith object on the list directly, without having to store the entire list and count through from the start.

  Alternatively, we may simply require a method of moving from each object to the next.
- We could also ask for a method for random generation of an object. If we have a technique for generating the ith object directly, we simply choose a random number in the range $\{0, \ldots, F(n) - 1\}$ and generate the corresponding object. If not, we have to rely on other methods such as Markov chains.

Here are a few examples. These will be considered in more detail in Chapter 3; it is not necessary to read what follows here in detail, but you are advised to skim through it.

Example: subsets The number of subsets of $\{1,\dots,n\}$ is 2^n. For each subset is specified by saying, for each number $i \in \{1,\dots,n\}$, whether i is in the subset or not; thus n binary choices are required to specify the subset.

Not only is this a simple formula to write down; it is easy to compute as well. It can clearly be done by starting with 1 and doubling n times (that is, n integer additions). Alternatively, it can be computed with at most $2\log_2 n$ integer multiplications. (In other words, we can choose to have fewer but more complicated operations.)

To see this, write n in base 2: $n = 2^{a_1} + 2^{a_2} + \cdots + 2^{a_r}$, where $a_1 > \cdots > a_r$. Now we can compute 2^{2^i} for $1 \le i \le a_1$ by a_1 successive squarings (noting that $2^{2^{i+1}} = \left(2^{2^i}\right)^2$); then $2^n = (2^{2^{a_1}}) \cdots (2^{2^{a_r}})$ requires $r-1$ further multiplications.

There is a simple recurrence relation for $F(n) = 2^n$, namely

$$F(0) = 1, \qquad F(n) = 2F(n-1) \text{ for } n \ge 1.$$

This expresses the calculation of 2^n by n doublings. Another recurrence relation, expressing the more efficient technique just outlined for computing 2^n, is given by

$$F(0) = 1, \qquad F(n) = \begin{cases} 2F(n-1) & \text{if } n \text{ is odd,} \\ F(n/2)^2 & \text{if } n \text{ is even.} \end{cases}$$

The ordinary generating function of the sequence (2^n) is

$$\sum_{n \ge 0} 2^n x^n = \frac{1}{1-2x},$$

while the exponential generating function is

$$\sum_{n \ge 0} \frac{2^n x^n}{n!} = \exp(2x).$$

(I will use $\exp(x)$ instead of e^x in these notes, except in some places involving calculus.)

No asymptotic estimate is needed, since we have a simple exact formula. Indeed, it is clear that 2^n is a number with $\lceil n \log_{10} 2 \rceil$ decimal digits.

Choosing a random subset, or generating all subsets in order, are easily achieved by the following method. For each $i \in \{0,\dots,2^n-1\}$, write i in base 2, producing a string of length n of zeros and ones. Now j belongs to the ith subset if and only if the jth symbol in the string is 1.

A procedure for moving from one set to the next can be produced using the *odometer principle*, based on the odometer or mileage gauge in a car. Represent a subset as above by a string of zeros and ones. To construct the next subset in the list, first identify the longest substring of ones at the end of the string. If the

string consists entirely of ones, then it is last in the order, and we have finished. Otherwise, this string is preceded by a zero; change the zero to a one, and the ones following it to zeros. For example, for $n = 3$, the odometer principle generates the strings

$$000, 001, 010, 011, 100, 101, 110, 111,$$

which correspond to the subsets

$$\emptyset, \{3\}, \{2\}, \{2,3\}, \{1\}, \{1,3\}, \{1,2\}, \{1,2,3\}$$

of $\{1,2,3\}$.

Notice that the binary strings are in *lexicographic order*, the order in which they would appear in a dictionary, regarding them as words over the alphabet $\{0,1\}$.

For $0 \leq k \leq n$, the number of k-element subsets of $\{1,\ldots,n\}$ is given by the *binomial coefficient*

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 1}.$$

The binomial coefficients are traditionally written in a triangular array where, for $n \geq 0$, the nth row contains the numbers $\binom{n}{0}$, $\binom{n}{1}$, $\ldots$, $\binom{n}{n}$. This is usually called *Pascal's triangle*, though, as we will see, it was not invented by Pascal. It begins like this:

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & 1 & & 1 & & & \\
 & & 1 & & 2 & & 1 & & \\
 & 1 & & 3 & & 3 & & 1 & \\
1 & & 4 & & 6 & & 4 & & 1
\end{array}$$

The most important property, and the reason for the name, is the form of the generating function for these numbers (regarded as a sequence indexed by k for fixed n), the *Binomial Theorem*:

$$\sum_{k=0}^{n} \binom{n}{k} x^k = (1+x)^n.$$

Example: permutations A *permutation* of the set $\{1,\ldots,n\}$ is a rearrangement of the elements of the set, that is, a bijective function from the set to itself. The number of permutations of $\{1,\ldots,n\}$ is the *factorial function* $n!$, defined as usual as the product of the natural numbers from 1 to n. This formula is not so satisfactory, involving an n-fold product. It can be expressed in other ways, as a sum:

$$n! = \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} (n-k)^n,$$

or as an integral:

$$n! = \int_0^\infty x^n e^{-x}\,dx.$$

Neither of these is easier to evaluate than the original definition. (We will meet both these formulae later on.)

The recurrence relation for $F(n) = n!$ is

$$F(0) = 1, \qquad F(n) = nF(n-1) \text{ for } n \geq 1.$$

This leads to the same method of evaluation as we saw earlier.

The ordinary generating function for $F(n) = n!$ fails to converge anywhere except at the origin. The exponential generating function is $1/(1-x)$, convergent for $|x| < 1$.

As an example to show that convergence is not necessary for a power series to be useful, let

$$\left(1 + \sum_{n\geq 1} n!x^n\right)^{-1} = 1 - \sum_{n\geq 1} c(n)x^n.$$

Then $c(n)$ is the number of connected permutations on $\{1,\ldots,n\}$. (A permutation π is *connected* if there does not exist k with $1 \leq k \leq n-1$ such that π maps $\{1,\ldots,k\}$ to itself.) This will be proved in the next chapter.

The approximate size of the factorial function is not obvious, as it was for powers of 2. An asymptotic estimate for $n!$ is given by *Stirling's formula*:

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

We give the proof later.

It is possible to generate permutations sequentially, or choose a random permutation, by a method similar to that for subsets, using a variable base.

The set of permutations of $\{1,\ldots,n\}$ forms a group under the operation of composition, the *symmetric group* of degree n, denoted by S_n.

Example: derangements A derangement is a permutation with no fixed points. Let $d(n)$ be the number of *derangements* of n.

There is a simple formula for $d(n)$: it is the nearest integer to $n!/e$. (This is one of the oldest formulae in combinatorics, having been proved by de Montmort in 1713.) This is also satisfactory as an asymptotic expression for $d(n)$; we can supplement it with the fact that $|d(n) - n!/e| < 1/(n+1)$ for $n > 0$.

This formula is not very good for calculation, since it requires accurate knowledge of e and operations of real (rather than integer) arithmetic. There are, however, two recurrence relations for $d(n)$; the second, especially, leads to efficient calculation:

$$d(0) = 1, d(1) = 0, \qquad d(n) = (n-1)(d(n-1) + d(n-2)) \text{ for } n \geq 2;$$
$$d(0) = 1, \qquad d(n) = nd(n-1) + (-1)^n \text{ for } n \geq 1.$$

The ordinary generating function for $d(n)$ fails to converge, but the exponential generating function is equal to $\exp(-x)/(1-x)$.

These facts will be proved in Chapter 4.

Since the probability that a random permutation is a derangement is about $1/\mathrm{e}$, we can choose a random derangement as follows: repeatedly choose a random permutation until a derangement is obtained. The expected number of choices necessary is about e.

Example: partitions The *partition number* $p(n)$ is the number of non-increasing sequences of positive integers with sum n. There is no simple formula for $p(n)$. However, quite a bit is known about it:

- The ordinary generating function is
$$\sum_{n\geq 0} p(n)x^n = \prod_{k\geq 1}(1-x^k)^{-1}.$$
- There is a recurrence relation:
$$p(n) = \sum(-1)^{k-1}p(n-k(3k-1)/2),$$
where the sum is over all non-zero values of k, positive and negative, for which $n-k(3k-1)/2 \geq 0$. Thus,
$$p(n) = p(n-1)+p(n-2)-p(n-5)-p(n-7)+p(n-12)+\cdots,$$
where there are about $\sqrt{8n/3}$ terms in the sum.

These facts will be proved in Chapter 4.

The asymptotics of $p(n)$ are rather complicated, and were worked out by Hardy, Littlewood, and Rademacher:
$$p(n) \sim \frac{1}{4n\sqrt{3}}\mathrm{e}^{\pi\sqrt{2n/3}}$$
(more precise estimates, including a convergent series representation, exist).

Example: set partitions The *Bell number* $B(n)$ is the number of partitions of the set $\{1,\ldots,n\}$. Again, no simple formula is known, and the asymptotics are very complicated. There is a recurrence relation,
$$B(n) = \sum_{k=1}^{n}\binom{n-1}{k-1}B(n-k),$$
and the exponential generating function is
$$\sum \frac{B(n)x^n}{n!} = \exp(\exp(x)-1).$$
Based on the recurrence one can derive a sequential generation algorithm, which calls itself recursively.

1.2 About how many?

As noted in the last section, if F and G are two functions on the natural numbers which do not vanish, we write $F \sim G$ if $F(n)/G(n) \to 1$ as $n \to \infty$. If $F(n)$ is the solution to a counting problem and $G(n)$ is a familiar analytic function, this tells us roughly the size of the collection we are counting. For example, we mentioned already that the number $n!$ of permutations of a set of size n is given approximately by *Stirling's formula*

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{\mathrm{e}}\right)^n.$$

Since $n!$ is the product of n numbers each at most n, it is clear that $n! \leq n^n$; the asymptotic formula gives a much more precise estimate.

In Chapter 11 we will also introduce further notation for *asymptotic analysis*, and in that chapter and the next we describe a variety of techniques for proving such estimates.

1.3 How hard is it?

A formula like 2^n (the number of subsets of an n-set) can be evaluated quickly for a given value of n. A more complicated formula with multiple sums and products will take longer to calculate. We could regard a formula which takes more time to evaluate than it would take to generate all the objects and count them as being useless in practice, even if it has theoretical value.

Traditional *computational complexity* theory refers to decision problems, where the answer is just 'yes' or 'no' (for example, 'Does this graph have a Hamiltonian circuit?'). The size of an instance of a problem is measured by the number of bits of data required to specify the problem (for example, $n(n-1)/2$ bits to specify a graph on n vertices). Then the time complexity of a problem is the function f, where $f(n)$ is the maximum number of steps required by a Turing machine to compute the answer for an instance of size n. (A *Turing machine* is a simple theoretical model of a computer which is capable of any theoretically-possible computation.)

To allow for variations in the format of the input data and in the exact specification of a Turing machine, complexity classes are defined with a broad brush: for example, P (or 'polynomial-time') consists of all problems whose time complexity is at most n^c for some constant c. (For more details, see Garey and Johnson, *Computers and Intractability*.)

For counting problems, the answer is a number rather than a single Boolean value (for example, 'How many Hamiltonian circuits does this graph have?'). Complexity theorists have defined the complexity class #P ('number-P') for this purpose.

Even this class is not really appropriate for counting problems of the type we mostly consider. Consider, for example, the question 'How many partitions does

an n-set have?' The input data is the integer n, which (if written in base 2) requires only $m = \lceil 1 + \log_2 n \rceil$ bits to specify. The question asks us to calculate the Bell number $B(n)$, which is greater than 2^{n-1} for $n > 2$, and so it takes time exponential in m simply to write down the answer! To get round this difficulty, it is usual to pretend that the size of the input data is actually n rather than $\log n$. (We can imagine that n is given by writing n consecutive 1s on the input tape of the Turing machine, that is, by writing n as a tally rather than in base 2.)

We have seen that computing 2^n (the number of subsets of an n-set) requires at most $\log n$ integer multiplications. But the integers may have as many as n digits, so each multiplication takes about n Turing machine steps. Similarly, the solution to a recurrence relation can be computed in time polynomial in n, provided that each individual computation can be.

On the other hand, a method which involves generating and testing every subset or permutation will take exponentially long, even if the generation and testing can be done efficiently.

A notion of complexity relevant to this situation is the polynomial delay model, which asks that the time required to generate each object should be at most n^c for some fixed c, even if the number of objects to be generated is greater than polynomial.

Of course, it is easy to produce combinatorial problems whose solution grows faster than, say, the exponential of a polynomial. For example, how many intersecting families of subsets of an n-set are there? The total number, for n odd, lies between $2^{2^{n-1}}$ and 2^{2^n}, so that even writing down the answer takes time exponential in n.

We will not consider complexity questions further in this book.

1.4 Exercises

1.1 Construct a bijection between the set of all k-element subsets of $\{1,\ldots,n\}$ containing no two consecutive elements, and the set of all k-element subsets of $\{1,\ldots,n-k+1\}$. Hence show that the number of such subsets is $\binom{n-k+1}{k}$.

When the UK National Lottery was introduced in 1994, the draw consisted of choosing six distinct numbers randomly from the set $\{1,\ldots,49\}$. What is the probability that the draw contained no two consecutive numbers?

1.2 (a) In Vancouver in 1984, I saw a Dutch pancake house advertised 'a thousand and one combinations' of toppings. What do you deduce?

(b) More recently McDonalds offered a meal deal with a choice from eight components of your meal, and advertised '40 312 combinations'. What do you deduce?

1.3 Prove the second formula for $n!$ given in the text:

$$n! = \int_0^\infty x^n \mathrm{e}^{-x}\,\mathrm{d}x.$$

1.4 Let $f(n)$ be the number of partitions of an n-set into parts of size 2.

(a) Prove that

$$f(n) = \begin{cases} 0 & \text{if } n \text{ is odd;} \\ 1 \cdot 3 \cdot 5 \cdots (n-1) & \text{if } n \text{ is even.} \end{cases}$$

(b) Prove that the exponential generating function for the sequence $(f(n))$ is $\exp(x^2)$.

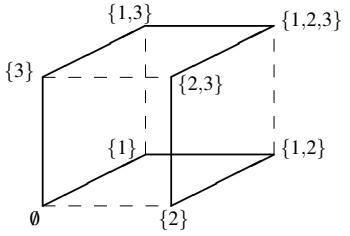
(c) Use Stirling's formula to prove that

$$f(n) \sim \sqrt{2}\left(\frac{2m}{\mathrm{e}}\right)^m$$

for $n = 2m$.

1.5 Show that it is possible to generate all subsets of $\{1,\ldots,n\}$ successively in such a way that each subset differs from its predecessor by the addition or removal of precisely one element. (Such a sequence is known as a *Gray code*.)

This picture might help.



Remark Gray codes are used in analog-to-digital converters. Since only one digit changes at a time when the input varies continuously, the damage caused by an error in reading the changing digit is minimised.

1.6 Counting can be used to prove structural results, as in the following exercise, which proves a theorem of Mantel: A graph with n vertices and more than $n^2/4$ edges must contain a triangle.

Consider a graph with n vertices, e edges and t triangles. Let x_i be the number of edges containing vertex i.

(a) Show by Inclusion–Exclusion that, if vertices i and j are joined, then at least $x_i + x_j - n$ triangles contain these two vertices.

(b) Hence show that

$$6t \geq \sum_i x_i^2 + \sum_j x_j^2 - 2ne.$$

(c) Use the Cauchy–Schwarz inequality to show that

$$\sum_i x_i^2 \geq 4e^2/n.$$

(d) Deduce that

$$t \geq \tfrac{1}{3}e(4e - n^2)/n.$$

(e) Hence show that, if $e > n^2/4$, the graph contains a triangle.

Remark The *Cauchy–Schwarz inequality* states that, if $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ are two sequences of real numbers, then

$$\left(\sum_{i=1}^n x_i y_i\right)^2 \leq \left(\sum_{i=1}^n x_i^2\right)\left(\sum_{i=1}^n y_i^2\right).$$

Geometrically, the norm of the vector $\vec{x} = (x_1, \ldots, x_n)$ is

$$\|\vec{x}\| = \left(\sum_{i=1}^n x_i^2\right)^{1/2}.$$

The Cauchy–Schwarz inequality says that the inner product of two vectors $\vec{x}$ and $\vec{y}$ cannot exceed in modulus the product of the norms of the vectors. Indeed, the ratio $(\vec{x} \cdot \vec{y})/\|x\| \cdot \|y\|$ is equal to the cosine of the angle between $\vec{x}$ and $\vec{y}$.

This is a remarkably useful inequality, in combinatorics as well as other branches of mathematics.

Can you prove the inequality? [**Hint:** Calculate the squared norm of the vector $\vec{x} + \lambda\vec{y}$; this is a quadratic function of λ which can never be negative.]

1.7 (a) Find an iterative method for listing the k-subsets of the natural numbers in *reverse lexicographic order* (that is, ordered by the largest element, and if largest elements are equal then by the second largest, and so on). Thus, for $k = 4$, the list begins

$$0123, 0124, 0134, 0234, 1234, 0125, \ldots$$

Your method should give an algorithm for moving from any subset to the next in the list.

(b) Show that, if $a_1 < \cdots < a_k$, then the position of $\{a_1, \ldots, a_k\}$ in the list is given by

$$\binom{a_1}{1} + \binom{a_2}{2} + \cdots + \binom{a_k}{k}.$$

Can you describe the inverse of this function, which enables us to write down the nth subset in the list?