Combinatorics

Peter Petersen

Summer 2007

Contents

Chapter 1

Counting

1.1 A General Combinatorial Problem

Instead of mostly focusing on the trees in the forest let us take an aerial view. You might get a bit of vertigo from this exposure, but the specific trees you have studied will hopefully come into sharper focus after the tour.

The treatment here was inspired by chapter 8 in Rademacher & Toeplitz, *The Enjoyment of Mathematics*.

We consider the problem of placing n objects into groups. We assume that there are k types of objects with i_1 of the first type, i_2 of the second etc. There are l groups, the first group can contain at most j_1 objects, the second at most j_2 etc. The number of ways of placing n objects with these constraints is denoted

$$\begin{bmatrix} i_1, ..., i_k \\ j_1, ..., j_l \end{bmatrix}_n$$

We think of taking n objects among the $i_1 + \cdots + i_k$ objects in the top row and then somehow placing and rearranging them among the groups in the second row. There is no general procedure for computing this number. Below we shall consider several special cases that correspond to what is covered in most texts.

There are different ways of interpreting the general problem. In the classical formulation the objects are marbles, the types colors and the groups urns where the differently colored marbles are placed. From a more modern perspective one might think of the types as being different manufacturing goods that are to be placed in various locales, repositories etc. Generally we shall speak of objects of various types to be placed in cells or groups. Quantum mechanics also allows for an interesting interpretation. The objects are now particles, perhaps before and after a collision that changes the attributes of the particles.

Keeping graph theory in mind we can come up with two more important pictures. Think of a bipartite (multi) graph with k vertices on the left and l on the right. We are allowed to have at most i_1 edges leaving the first vertex on the left, i_2 from the second etc. Likewise we allow at most j_1 edges to meet the

first vertex on the right, j_2 going to the second etc. Alternately we can create a network. There will be one source, from this source there are k outgoing edges, the first has capacity i_1, the second capacity i_2, etc. The k vertices at the ends of these edges are then connected to l vertices in a complete bipartite graph $K_{k,l}$ where each edge has ∞ capacity. The right l vertices are then connected to a sink, where the first edge has capacity j_1, the second j_2, etc.

By reversing the direction in all of the above examples we get an important symmetry property

$$\begin{bmatrix} i_1, ..., i_k \\ j_1, ..., j_l \end{bmatrix}_n = \begin{bmatrix} j_1, ..., j_l \\ i_1, ..., i_k \end{bmatrix}_n$$

Thus types and groups are interchangeable. Essentially what we are doing in moving from types to groups is to reassign types. Taking painted balls and then repainting them according to the groups were they get placed.

Note that

$$\begin{bmatrix} i_1, ..., i_k \\ j_1, ..., j_l \end{bmatrix}_n = 0$$

if $n > i_1 + \cdots + i_k$ or if $n > j_1 + \cdots + j_l$. Otherwise there should always be something to count.

1.2 Arrangements

An *arrangement* is a problem of the above type where at most one object can be placed in each group. In this case we often refer to groups as cells. Thus $j_1 = \cdots = j_l = 1$. We can solve almost all problems of this kind using a variety of tricks. The most important is to use recurrence or induction on the number of cells.

The basic arrangement is a permutation, where we have n types of objects that are placed in n different locations. The total number of such permutations is denoted

$$P(n) = \begin{bmatrix} 1_1, ..., 1_n \\ 1_1, ..., 1_n \end{bmatrix}_n$$

We observe that there are n possibilities for putting an object in the last cell

$$\begin{aligned} P(n) &= nP(n-1) \\ \begin{bmatrix} 1_1, ..., 1_n \\ 1_1, ..., 1_n \end{bmatrix}_n &= n \begin{bmatrix} 1_1, ..., 1_{n-1} \\ 1_1, ..., 1_{n-1} \end{bmatrix}_{n-1} \end{aligned}$$

The last formula seems a bit disturbing as we seem to have assumed that the last object was placed in the last cell. However, as we only numbered the types without attaching any real meaning to them we are allowed to reassign the meaning of the types after one object has been placed in the last cell. This kind of blasé use of symbols will be employed without discussion. It is precisely what makes math easier rather than harder! Just consider our reversal of types and groups above completely throwing to the winds our interpretation of the original problem.

Repeating the recurrence n times gives us

$$P(n) = \begin{bmatrix} 1_1, ..., 1_n \\ 1_1, ..., 1_n \end{bmatrix}_n = n!$$

We now restrict the permutation by decreasing the number of cells $l < k$

$$P(k,l) = \begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_l$$

This is often interpreted as counting the number of words that be created with a large alphabet of distinct letters. We see again that

$$\begin{array}{rcl} P(k,l) & = & kP(k-1,l-1) \\ \begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_l & = & k \begin{bmatrix} 1_1, ..., 1_{k-1} \\ 1_1, ..., 1_{l-1} \end{bmatrix}_{l-1} \end{array}$$

as there are k objects that can go in the last cell. Repeating the recurrence l times gives us

$$P(k,l) = \begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_l = k(k-1)\cdots(k-(l-1)) = \frac{k!}{(k-l)!} = k_{(l)}.$$

There is also a proof of this identity that doesn't use recurrence. Any permutation of k objects can be achieved by first arranging l of the objects and then the remaining $k-l$ in the $k-l$ cells left over. Thus

$$\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_k \end{bmatrix}_k = \begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_l \begin{bmatrix} 1_1, ..., 1_{k-l} \\ 1_1, ..., 1_{k-l} \end{bmatrix}_{k-l}$$

In other words:

$$\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_l = \frac{\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_k \end{bmatrix}_k}{\begin{bmatrix} 1_1, ..., 1_{k-l} \\ 1_1, ..., 1_{k-l} \end{bmatrix}_{k-l}} = \frac{k!}{(k-l)!}$$

This type of over counting comes in handy in several situations below.

Next consider the symmetric case

$$\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_k$$

where $k < l$. This is like placing all letters in an alphabet in a sentence where the spaces correspond to the cells not being used. The symmetry condition gives

$$\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_k = \begin{bmatrix} 1_1, ..., 1_l \\ 1_1, ..., 1_k \end{bmatrix}_k = \frac{l!}{(l-k)!}$$

The general arrangement

$$\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_n$$

can be realized by first selecting n types and then placing them among the l cells:

$$\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_n \end{bmatrix}_n \begin{bmatrix} 1_1, ..., 1_n \\ 1_1, ..., 1_l \end{bmatrix}_n$$

However, this over counts as we have then also included permutations of the n selected objects. This shows that

$$\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_n \begin{bmatrix} 1_1, ..., 1_n \\ 1_1, ..., 1_n \end{bmatrix}_n = \begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_n \end{bmatrix}_n \begin{bmatrix} 1_1, ..., 1_n \\ 1_1, ..., 1_l \end{bmatrix}_n$$

and

$$\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_n = \frac{\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_n \end{bmatrix}_n \begin{bmatrix} 1_1, ..., 1_n \\ 1_1, ..., 1_l \end{bmatrix}_n}{\begin{bmatrix} 1_1, ..., 1_n \\ 1_1, ..., 1_n \end{bmatrix}_n} = \frac{k!l!}{(k-n)!\,(l-n)!n!}.$$

A similar argument that doesn't use over counting will be given below.

We next consider arrangements with (unlimited) *replacement*. This means that there is an infinite supply of each type. The arrangements where we have two types are

$$\begin{bmatrix} \infty, \infty \\ 1_1, ..., 1_l \end{bmatrix}_l$$

and more generally k types

$$\begin{bmatrix} \infty_1, ..., \infty_k \\ 1_1, ..., 1_l \end{bmatrix}_l.$$

We are thus creating words or numbers where we are allowed to reuse letters or digits. In the first case we are counting the number of binary numbers with l digits and in the general case l digit numbers in base k. We can again use recurrence by noting that are k possibilities for what to place in the l^{th} cell.

$$\begin{bmatrix} \infty_1, ..., \infty_k \\ 1_1, ..., 1_l \end{bmatrix}_l = k \begin{bmatrix} \infty_1, ..., \infty_k \\ 1_1, ..., 1_{l-1} \end{bmatrix}_{l-1}$$

since this didn't depend on l at all we can repeat it l times to get

$$\begin{bmatrix} \infty_1, ..., \infty_k \\ 1_1, ..., 1_l \end{bmatrix}_l = k^l.$$

Finally we treat a problem were limited replacement is allowed. We wish to study how one can *rearrange* a specific string of types such as the letters in MISSISSIPPI. This is the same as counting

$$\begin{bmatrix} 1_M, 4_I, 4_S, 2_P \\ 1_1, ..., 1_{11} \end{bmatrix}_{11}$$

More generally we are considering

$$\begin{bmatrix} i_1, ..., i_k \\ 1_1, ..., 1_l \end{bmatrix}_l$$

where $i_1 + \cdots + i_k = l$.

If we think of a permutation of l letters and then group these letters according to type, then this permutation can be obtained by first identifying how many ways there are of arranging the objects and then performing permutations of the objects of a given type. Specifically

$$\begin{bmatrix} 1_1, ..., 1_l \\ 1_1, ..., 1_l \end{bmatrix}_l = \begin{bmatrix} i_1, ..., i_k \\ 1_1, ..., 1_l \end{bmatrix}_l \begin{bmatrix} 1_1, ..., 1_{i_1} \\ 1_1, ..., 1_{i_1} \end{bmatrix}_{i_1} \cdots \begin{bmatrix} 1_1, ..., 1_{i_k} \\ 1_1, ..., 1_{i_k} \end{bmatrix}_{i_k}$$

This shows that

$$\begin{bmatrix} i_1, ..., i_k \\ 1_1, ..., 1_l \end{bmatrix}_l = \frac{l!}{i_1! \cdots i_k!} = \binom{l}{i_1, ..., i_k}.$$

An important special case occurs when we take n identical objects and place them in l cells. If we regard the empty $l-n$ cells as being populated by a second type we have

$$\begin{bmatrix} n \\ 1_1, ..., 1_l \end{bmatrix}_n = \begin{bmatrix} n, l-n \\ 1_1, ..., 1_l \end{bmatrix}_l = \binom{l}{n, l-n}.$$

1.3 Combinations

In a *combination* we take objects of different types and then place them in one group. Thus we are considering

$$\begin{bmatrix} i_1, ..., i_k \\ n \end{bmatrix}_n$$

The basic case is when the objects are distinct:

$$C(k,l) = \begin{bmatrix} 1_1, ..., 1_k \\ l \end{bmatrix}_l = \begin{bmatrix} l \\ 1_1, ..., 1_k \end{bmatrix}_l = \binom{k}{l, k-l} = \binom{k}{l}$$

We could also use arrangements to count as follows

$$\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_l = \begin{bmatrix} 1_1, ..., 1_k \\ l \end{bmatrix}_l \begin{bmatrix} 1_1, ..., 1_l \\ 1_1, ..., 1_l \end{bmatrix}_l$$

so

$$\begin{bmatrix} 1_1, ..., 1_k \\ l \end{bmatrix}_l = \frac{k!}{(k-l)!l!}$$

In other words a permutation of l elements out of a collection of k objects can be constructed by first selecting the objects (the combination) and then permuting them.

The symmetric problem

$$\begin{bmatrix} l \\ 1_1, ..., 1_k \end{bmatrix}_l$$

is also often called a combination even though we are obviously arranging identical objects.

The general arrangement where $n < \min\{k, l\}$ can now be calculated by first selecting the objects that we use (a combination) and then placing them in cells

$$\begin{aligned}\begin{bmatrix}1_1, ..., 1_k\\ 1_1, ..., 1_l\end{bmatrix}_n &= \begin{bmatrix}1_1, ..., 1_k\\ n\end{bmatrix}_n \begin{bmatrix}1_1, ..., 1_l\\ 1_1, ..., 1_n\end{bmatrix}_n \\ &= \binom{k}{n}\frac{l!}{(l-n)!} \\ &= \frac{k!l!}{(k-n)!\,(l-n)!n!}\end{aligned}$$

Multi-combinations consist of taking distinct objects and then putting them in several groups:

$$\begin{bmatrix}1_1, ..., 1_k\\ j_1, ..., j_l\end{bmatrix}_k$$

with $k = j_1 + \cdots + j_l$. Symmetrizing gives us an arrangement

$$\begin{bmatrix}1_1, ..., 1_k\\ j_1, ..., j_l\end{bmatrix}_k = \begin{bmatrix}j_1, ..., j_l\\ 1_1, ..., 1_k\end{bmatrix}_k = \binom{k}{j_1, ..., j_l}.$$

These coefficients come about when we do a multinomial expansion

$$(x_1 + \cdots + x_l)^k = \sum_{j_1,...,j_l} \binom{k}{j_1, ..., j_l} x_1^{j_1} \cdots x_l^{j_l}.$$

Finally we turn our attention to the fairly tricky situation of combinations with unlimited replacement. These are the simplest types of combinations where arrangements don't immediately yield the answer. We are considering

$$\begin{bmatrix}\infty_1, ..., \infty_k\\ n\end{bmatrix}_n = \left(\binom{k}{n}\right).$$

By symmetrizing we can reinterpret this as dividing n identical objects into k groups, i.e., the number

$$\begin{bmatrix}n\\ \infty_1, ..., \infty_k\end{bmatrix}_n$$

Suppose, e.g., that there are 8 identical xs to be divided into 5 groups. Such a division might look like

$$xx, xxx, , xxx,$$

if the third and fifth groups are empty, or like

$$, xx, x, xxxx, x$$

if the first group is empty. In general there are $n+k-1$ symbols in this string: n letters and $k-1$ commas to divide the letters into groups. A grouping is then determined if we place either the letters or the commas. Thus

$$\left(\binom{k}{n}\right) = \begin{bmatrix} n \\ \infty_1, ..., \infty_k \end{bmatrix}_n = \binom{n+k-1}{n} = \binom{n+k-1}{k-1}.$$

Combination with replacement comes up in a very interesting context. Suppose we have a function $f : \mathbb{R}^k \to \mathbb{R}$ such that f has partial derivatives of all orders. How many different partial derivatives are there of order n? Such a partial derivative looks like

$$\begin{array}{rcl} & & \dfrac{\partial^n f}{\partial x_1^{n_1} \cdots \partial x_k^{n_k}}, \\ n & = & n_1 + \cdots + n_k \end{array}$$

since the order in which we take the derivatives is irrelevant. Thus we are simply asking about the number of ways of distributing n objects into k groups.

1.4 Nomenclature

At this point we should make a few translations so as to connect this more theoretical view with some of the terms that are often used in combinatorics.

One often uses the terms *distinguishable* and *indistinguishable* for the types as well as the groups. But this also supposes that the division into types and groups is extremely clear.

Distributing n out of k distinguishable objects into l different distinguishable cells is the same as counting

$$\begin{bmatrix} 1_1, ..., 1_k \\ 1_1, ..., 1_l \end{bmatrix}_n$$

In other words it is a fairly simple arrangement. When objects get identified according to type and groups/cells are allowed to contain several objects we are considering

$$\begin{bmatrix} i_1, ..., i_k \\ j_1, ..., j_l \end{bmatrix}_n$$

Distributing indistinguishable objects into distinguishable cells is the same as having just one type that is being arranged

$$\begin{bmatrix} k \\ 1_1, ..., 1_l \end{bmatrix}_n$$

By symmetry this is also a combination

$$\begin{bmatrix} k \\ 1_1, ..., 1_l \end{bmatrix}_n = \begin{bmatrix} 1_1, ..., 1_l \\ k \end{bmatrix}_n$$

Therefore, problems with indistinguishable objects are often referred to as combinations even if they appear to be arrangements.

The symmetry and how it allows us to reinterpret the same problem in two ways is often confusing. This is why we will stick to the more formal approach of types and groups/cells.

1.5 More Restrictions

It is possible to introduce further restrictions. One might require that a certain number of objects of each type are being used, or that a certain number of objects are placed in each group. Symmetry tells us that these two conditions are equivalent, but they could also be imposed simultaneously.

The most general set-up is as follows. We have sets of types $A_1, ..., A_k$ as well as sets of groups $B_1, ..., B_l$. These sets consists of nonnegative integers. In the scenario considered above $A_1 = \{0, 1, ..., i_1\}$ indicating that we can take any number of objects from A_1 as long as it doesn't exceed i_1. Now we allow for the sets to look like

$$\begin{aligned} A_1 &= \{0, 3, 6, 7\} \\ A_2 &= \{2n | n = 0, 1, 2, ...\} \\ A_3 &= \{p | p \text{ is a prime}\} \\ &\quad etc \end{aligned}$$

Similarly the groups can have wild restrictions on how many objects they contain.

We then seek to find the number of ways of selecting n objects from $A_1, ..., A_k$ and placing them in $B_1, ..., B_l$ according to these constraints. This number is denoted

$$\begin{bmatrix} A_1, ..., A_k \\ B_1, ..., B_l \end{bmatrix}_n.$$

There might now be some very subtle conditions on n for such a placement to be possible. If, e.g., all types have to be selected in quantities that are even, then n must also be even.

As it stands, this should be even trickier to calculate than

$$\begin{bmatrix} i_1, ..., i_k \\ j_1, ..., j_l \end{bmatrix}_n$$

However it turns out that there are useful reductions when we consider either combinations or arrangements were each cell is used exactly once. Specifically we will consider the general combination

$$\begin{bmatrix} A_1, ..., A_k \\ n \end{bmatrix}_n$$

and the general arrangement

$$\begin{bmatrix} A_1, ..., A_k \\ 1_1, ..., 1_n \end{bmatrix}_n$$

where we select n objects of various types and in quantities specified by type and either just combine them in one group or arrange them in n different locations.

We consider combinations first as they turn out to be slightly simpler. The most efficient recurrence is based on reducing the number of types. So we check what happens if we select $r \leq n$ objects from A_k and the remaining $n - r$ from $A_1, ..., A_{k-1}$. In other words we note that

$$\begin{bmatrix} A_1, ..., A_k \\ n \end{bmatrix}_n = \sum_{r=0}^{n} \begin{bmatrix} A_1, ..., A_{k-1} \\ n-r \end{bmatrix}_{n-r} \begin{bmatrix} A_k \\ r \end{bmatrix}_r$$

where

$$\begin{bmatrix} A_k \\ r \end{bmatrix}_r = \begin{cases} 1 & \text{if } r \in A_k \\ 0 & \text{if } r \notin A_k \end{cases}$$

Thus the sum can be rewritten as

$$\begin{bmatrix} A_1, ..., A_k \\ n \end{bmatrix}_n = \sum_{r \in A_k} \begin{bmatrix} A_1, ..., A_{k-1} \\ n-r \end{bmatrix}_{n-r}$$

For the basic combination this looks like

$$\begin{aligned} \begin{bmatrix} 1_1, 1_2, ..., 1_k \\ n \end{bmatrix}_n &= \sum_{r=0,1} \begin{bmatrix} 1_1, ..., 1_{k-1} \\ n-r \end{bmatrix}_{n-r} \\ &= \begin{bmatrix} 1_1, ..., 1_{k-1} \\ n \end{bmatrix}_n + \begin{bmatrix} 1_1, ..., 1_{k-1} \\ n-1 \end{bmatrix}_{n-1} \end{aligned}$$

as we either don't pick or pick the element of the last type. In more classical language this is Pascal's formula

$$\binom{k}{n} = \binom{k-1}{n} + \binom{k-1}{n}.$$

All in all we have a recurrence method for finding very general combinations, where the number of types is reduced each time we use the recurrence.

More generally we obtain the formula

$$\begin{bmatrix} A_1, ..., A_k \\ n \end{bmatrix}_n = \sum_{n_1 + \cdots + n_k = n} \begin{bmatrix} A_1 \\ n_1 \end{bmatrix}_{n_1} \cdots \begin{bmatrix} A_k \\ n_k \end{bmatrix}_{n_k}$$

by counting over all possible $n_1, ..., n_k$ that add up to n and checking if we can extract n_1 from A_1 etc. This is simply a more general version of the recurrence where we have spelled out all possibilities of creating the combinations on the left-hand side.

Arrangements work in a similar fashion. But we must be more careful. The idea is that if we select r objects from A_k, then there are $\binom{n}{r}$ ways of placing these identical objects among the n cells. The remaining $n-r$ objects from the $k-1$ types are then arranged among the remaning $n-r$ cells. This leads to the recurrence

$$\begin{aligned}\begin{bmatrix} A_1, ..., A_k \\ 1_1, ..., 1_n \end{bmatrix}_n &= \sum_{r=0}^{n} \binom{n}{r} \begin{bmatrix} A_1, ..., A_{k-1} \\ 1_1, ..., 1_{n-r} \end{bmatrix}_{n-r} \begin{bmatrix} A_k \\ r \end{bmatrix}_r \\ &= \sum_{r \in A_k} \binom{n}{r} \begin{bmatrix} A_1, ..., A_{k-1} \\ 1_1, ..., 1_{n-r} \end{bmatrix}_{n-r}\end{aligned}$$

If instead of calculating $\begin{bmatrix} A_1, ..., A_k \\ 1_1, ..., 1_n \end{bmatrix}_n$ we consider the normalized quantity,

$$\frac{1}{n!} \begin{bmatrix} A_1, ..., A_k \\ 1_1, ..., 1_n \end{bmatrix}_n$$

then we get a formula that is almost like the combination recurrence

$$\frac{1}{n!} \begin{bmatrix} A_1, ..., A_k \\ 1_1, ..., 1_n \end{bmatrix}_n = \sum_{r=0}^{n} \left(\frac{1}{(n-r)!} \begin{bmatrix} A_1, ..., A_{k-1} \\ 1_1, ..., 1_{n-r} \end{bmatrix}_{n-r} \right) \left(\frac{1}{r!} \begin{bmatrix} A_k \\ r \end{bmatrix}_r \right)$$

We also quickly obtain the more general version

$$\begin{bmatrix} A_1, ..., A_k \\ 1_1, ..., 1_n \end{bmatrix}_n = \sum_{n_1 + \cdots + n_k = n} \binom{n}{n_1, ..., n_k} \begin{bmatrix} A_1 \\ n_1 \end{bmatrix}_{n_1} \cdots \begin{bmatrix} A_k \\ n_k \end{bmatrix}_{n_k}$$

by adding up over all possible ways of extracting n objects and then counting the number of ways of distributing them in the cells given that there are n_1 of type 1 etc. This formula also becomes more symmetric if we use factorial normalization

$$\frac{1}{n!} \begin{bmatrix} A_1, ..., A_k \\ 1_1, ..., 1_n \end{bmatrix}_n = \sum_{n_1 + \cdots + n_k = n} \left(\frac{1}{n_1!} \begin{bmatrix} A_1 \\ n_1 \end{bmatrix}_{n_1} \right) \cdots \left(\frac{1}{n_k!} \begin{bmatrix} A_k \\ n_k \end{bmatrix}_{n_k} \right)$$

These recurrence formulas can be incorporated into a different accounting method in a surprising fashion.

Chapter 2

Generating Functions

2.1 Generating Functions for Combinations

The problem is to calculate

$$c_n = \begin{bmatrix} A_1, ..., A_k \\ n \end{bmatrix}_n$$

for all values of n. To accomplish this we construct the *generating function*

$$\begin{aligned} \sum_{n=0}^{\infty} \begin{bmatrix} A_1, ..., A_k \\ n \end{bmatrix}_n x^n &= \sum_{n=0}^{\infty} c_n x^n \\ &= c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n + \cdots \end{aligned}$$

This is a formal infinite sum. No convergence is implied.

The generating function for just one type, say A_k, looks like

$$\sum_{n=0}^{\infty} \begin{bmatrix} A_k \\ n \end{bmatrix}_n x^n = \sum_{n \in A_n} x^n$$

Thus we can select n objects in 1 way if A_k allows that. Here are a few examples of generating functions for just one type

$$\begin{aligned} &1 + x, \\ &1 + x + x^2 + x^3 + \cdots, \\ &x^r + x^{r+1} + x^{r+2} + \cdots, \\ &x + x^3 + x^5 + \cdots, \\ &x^2 + x^3 + x^5 + x^7 + x^{11} + \cdots \end{aligned}$$

Note that the only coefficients we see are 1, the terms that don't appear are the ones with coefficient 0 and correspond to selections that the type does NOT

allow. In the first case the type allows for 0 or 1 object to be selected. In the second case any number of objects. In the third case at least r objects. In the fourth case only an odd number of objects. Finally the last case is when the type allows a number of objects that is a prime number.

Going back to our general problem we define the generating function for each type as

$$\sum_{n=0}^{\infty} \begin{bmatrix} A_i \\ n \end{bmatrix}_n x^n = c_0^i + c_1^i x + c_2^i x^2 + \cdots + c_n^i x^n + \cdots .$$

The claim is that the original generating function is the product of the generating functions for the types:

$$\sum_{n=0}^{\infty} \begin{bmatrix} A_1, ..., A_k \\ n \end{bmatrix}_n x^n = \prod_{i=1}^{k} \left(\sum_{n=0}^{\infty} \begin{bmatrix} A_i \\ n \end{bmatrix}_n x^n \right)$$

or

$$\begin{aligned} \sum_{n=0}^{\infty} c_n x^n &= \prod_{i=1}^{k} \left(\sum_{n=0}^{\infty} c_n^i x^n \right) \\ &= \left(\sum_{n=0}^{\infty} c_n^1 x^n \right) \left(\sum_{n=0}^{\infty} c_n^2 x^n \right) \cdots \left(\sum_{n=0}^{\infty} c_n^k x^n \right). \end{aligned}$$

The product on the right can be multiplied out more easily if we use different indices for n in each sum. When manipulating such sums you should work with them as if they were multiple integrals with $n_1, ..., n_k$ being the variables and be happy that no infinitesimals are messing up the calculations.

$$\begin{aligned} \text{RHS} &= \left(\sum_{n_1=0}^{\infty} c_{n_1}^1 x^{n_1} \right) \left(\sum_{n_2=0}^{\infty} c_{n_2}^2 x^{n_2} \right) \cdots \left(\sum_{n_k=0}^{\infty} c_{n_k}^k x^{n_k} \right) \\ &= \sum_{n_1,...,n_k=0}^{\infty} \left(c_{n_1}^1 c_{n_2}^2 \cdots c_{n_k}^k \right) x^{n_1+n_2+\cdots+n_k} \\ &= \sum_{n=0}^{\infty} \left(\sum_{n_1+\cdots+n_k=n} c_{n_1}^1 c_{n_2}^2 \cdots c_{n_k}^k \right) x^n \end{aligned}$$

Thus we must show that

$$c_n = \sum_{n_1+\cdots+n_k=n} c_{n_1}^1 c_{n_2}^2 \cdots c_{n_k}^k$$

but this is simply our recurrence formula

$$\begin{bmatrix} A_1, ..., A_k \\ n \end{bmatrix}_n = \sum_{n_1+\cdots+n_k=n} \begin{bmatrix} A_1 \\ n_1 \end{bmatrix}_{n_1} \cdots \begin{bmatrix} A_k \\ n_k \end{bmatrix}_{n_k}$$

At this point we seem to have achieved absolutely nothing. When we get to the examples below we shall see that the advantage of this method is that we really can work with generating functions as functions.

2.2 Exponential Generating Functions

A similar strategy can be used when dealing with arrangements

$$\begin{bmatrix} A_1, ..., A_k \\ 1_1, ..., 1_n \end{bmatrix}_n$$

The difference is that to get the recurrence to work as above we have to use the factorial normalization. Thus we consider the generating functions

$$\sum_{n=0}^{\infty} \frac{1}{n!} \begin{bmatrix} A_1, ..., A_k \\ 1_1, ..., 1_n \end{bmatrix}_n x^n = \sum_{n=0}^{\infty} p_n \frac{x^n}{n!}$$

Such generating functions are called *exponential generating functions.*

For just one type, say A_k, we get

$$\sum_{n=0}^{\infty} \frac{1}{n!} \begin{bmatrix} A_k \\ n \end{bmatrix}_n x^n = \sum_{n=0}^{\infty} p_n^k \frac{x^n}{n!}$$

as

$$\begin{bmatrix} A_k \\ 1_1, ..., 1_n \end{bmatrix}_n = \begin{bmatrix} A_k \\ n \end{bmatrix}_n = \begin{cases} 1 & \text{if } n \in A_k \\ 0 & \text{if } n \notin A_k \end{cases}$$

The above examples, as exponential generating functions, will look like

$$\begin{aligned} & 1 + x, \\ & 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots, \\ & \frac{x^r}{r!} + \frac{x^{r+1}}{(r+1)!} + \frac{x^{r+2}}{(r+2)!} + \cdots, \\ & x + \frac{x^3}{6} + \frac{x^5}{5!} + \cdots, \\ & \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^5}{5!} + \frac{x^7}{7!} + \frac{x^{11}}{11!} + \cdots \end{aligned}$$

The grand formula now asserts that

$$\sum_{n=0}^{\infty} \begin{bmatrix} A_1, ..., A_k \\ 1_1, ..., 1_n \end{bmatrix}_n \frac{x^r}{n!} = \prod_{i=0}^{k} \left(\sum_{n=0}^{\infty} \begin{bmatrix} A_i \\ n \end{bmatrix}_n \frac{x^n}{n!} \right)$$

or

$$\begin{aligned} \sum_{n=0}^{\infty} p_n \frac{x^r}{n!} &= \prod_{i=0}^{k} \left(\sum_{n=0}^{\infty} p_n^i \frac{x^n}{n!} \right) \\ &= \left(\sum_{n=0}^{\infty} p_n^1 \frac{x^n}{n!} \right) \left(\sum_{n=0}^{\infty} p_n^2 \frac{x^n}{n!} \right) \cdots \left(\sum_{n=0}^{\infty} p_n^k \frac{x^n}{n!} \right). \end{aligned}$$

The proof is basically the same

$$\begin{aligned}\text{RHS} &= \left(\sum_{n_1=0}^{\infty} p_{n_1}^1 \frac{x^{n_1}}{n_1!}\right)\left(\sum_{n_2=0}^{\infty} p_{n_2}^2 \frac{x^{n_2}}{n_2!}\right)\cdots\left(\sum_{n_k=0}^{\infty} p_{n_k}^k \frac{x^{n_k}}{n_k!}\right)\\ &= \sum_{n_1,\ldots,n_k=0}^{\infty} \left(p_{n_1}^1 p_{n_2}^2 \cdots p_{n_k}^k\right) \frac{x^{n_1+n_2+\cdots+n_k}}{n_1!n_2!\cdots n_k!}\\ &= \sum_{n=0}^{\infty}\left(\sum_{n_1+\cdots+n_k=n} \binom{n}{n_1, n_2, ..., n_k} p_{n_1}^1 p_{n_2}^2 \cdots p_{n_k}^k\right)\frac{x^n}{n!}\end{aligned}$$

Thus we have to show that

$$p_n = \sum_{n_1+\cdots+n_k=n} \binom{n}{n_1, n_2, ..., n_k} p_{n_1}^1 p_{n_2}^2 \cdots p_{n_k}^k$$

but this is the formula

$$\begin{bmatrix} A_1, ..., A_k \\ 1_1, ..., 1_n \end{bmatrix}_n = \sum_{n_1+\cdots+n_k=n} \binom{n}{n_1, ..., n_k} \begin{bmatrix} A_1 \\ n_1 \end{bmatrix}_{n_1} \cdots \begin{bmatrix} A_k \\ n_k \end{bmatrix}_{n_k}$$

2.3 Examples

First we try a few simple examples were the answers are known.

Suppose that each type allows you select at most one object. This means that both the ordinary and exponential generating function for a type is $1 + x$. Upon multiplying we obtain

$$(1+x)^k = 1 + \binom{k}{1}x + \binom{k}{2}x^2 + \cdots + \binom{k}{n}x^n + \cdots + \binom{k}{k}x^k.$$

This is the generating function in case we form a combination. Note that when $n > k$ we have $c_n = 0$ corresponding to the fact that we can't have more objects than types.

In case we want to arrange the objects we still have to multiply as above, but then we also have to put it in the form of an exponential generating function.

$$\begin{aligned}(1+x)^k &= 1 + \binom{k}{1}x + \binom{k}{2}x^2 + \cdots + \binom{k}{n}x^n + \cdots + \binom{k}{k}x^k\\ &= 1 + \binom{k}{1}x + 2\binom{k}{2}\frac{x^2}{2} + \cdots + n!\binom{k}{n}\frac{x^n}{n!} + \cdots + k!\binom{k}{k}\frac{x^k}{k!}.\end{aligned}$$

Thus there are

$$n!\binom{k}{n} = \frac{k!}{(k-n)!} = k_{(n)}$$

ways of arranging n distinct objects of k types.

If there are no restrictions on each type we get

$$\begin{aligned}\sum_{n=0}^{\infty} c_n x^n &= \left(1+x+x^2+\cdots\right)^k \\ &= \left(\frac{1}{1-x}\right)^k \\ &= (1-x)^{-k} \\ &= \sum_{n=0}^{\infty} \binom{-k}{n} (-x)^n \\ &= \sum_{n=0}^{\infty} (-1)^n \binom{-k}{n} x^n \\ &= \sum_{n=0}^{\infty} \left(\binom{k}{n}\right) x^n\end{aligned}$$

This formula is obtained by computing the Taylor formula for

$$f(x) = (1+x)^{\alpha}$$

The derivatives are

$$\begin{aligned}f' &= \alpha (1+x)^{\alpha-1}, \\ f'' &= \alpha(\alpha-1)(1+x)^{\alpha-2}, ...\end{aligned}$$

So

$$\begin{aligned}f'(0) &= \alpha, \\ f''(0) &= \alpha(\alpha-1), \\ &\vdots \\ f^{(n)}(0) &= \alpha(\alpha-1)\cdots(\alpha-(n-1))\end{aligned}$$

and

$$\begin{aligned}(1+x)^{\alpha} &= \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n \\ &= \sum_{n=0}^{\infty} \frac{\alpha(\alpha-1)\cdots(\alpha-(n-1))}{n!} x^n \\ &= \sum_{n=0}^{\infty} \binom{\alpha}{n} x^n\end{aligned}$$

Mirroring the combinatorial formula

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

Newton was the first to discover this remarkable generalization of the binomial theorem, where α can be any real number whatsoever. Some serious calculus has to be invoked in order to prove this formula as the right-hand side is only a formal sum. Next we note the important relation

$$\begin{aligned}
(-1)^n \binom{-k}{n} &= (-1)^n \frac{(-k)(-k-1)\cdots(-k-(n-1))}{n!} \\
&= (-1)^n (-1)^n \frac{(k)(k+1)\cdots(k+(n-1))}{n!} \\
&= \frac{(k+n-1)\cdots(k+1)(k)}{n!} \\
&= \binom{k+n-1}{n} \\
&= \left(\binom{k}{n}\right)
\end{aligned}$$

With exponential generating functions we get instead

$$\begin{aligned}
\sum_{n=0}^{\infty} p_n \frac{x^n}{n!} &= \left(1 + x + \frac{x^2}{2} + \cdots\right)^k \\
&= \left(e^x\right)^k \\
&= e^{kx} \\
&= \sum_{n=0}^{\infty} \frac{(kx)^n}{n!} \\
&= \sum_{n=0}^{\infty} k^n \frac{x^n}{n!}.
\end{aligned}$$

Next let us try to have just two types with the first allowing an even number of objects and the other an odd number.

The combination problem is then solved by

$$\begin{aligned}
\left(1 + x^2 + x^4 + \cdots\right)\left(x + x^3 + x^5 + \cdots\right) &= \left(1 + x^2 + x^4 + \cdots\right) x \left(1 + x^2 + x^4 + \cdots\right) \\
&= x\left(1 + x^2 + x^4 + \cdots\right)^2 \\
&= x\left(\frac{1}{1-x^2}\right)^2 \\
&= x\left(1 - x^2\right)^{-2} \\
&= x \sum_{n=0}^{\infty} \left(\binom{2}{n}\right) \left(x^2\right)^n \\
&= \sum_{n=0}^{\infty} \left(\binom{2}{n}\right) x^{2n+1} \\
&= \sum_{n=0}^{\infty} (n+1)\, x^{2n+1}
\end{aligned}$$

Thus we have to combine an odd number of objects, namely, $2n+1$ and this can be done in $n+1$ ways. Let us see if this makes sense. The type that allows an odd number of objects is $\{1,3,5,...,2n+1\}$ and the rest come from the other type. Since there are $n+1$ odd numbers between 1 and $2n+1$ we have justified the formula. We have also seen that while the generating function approach can be used it isn't necessarily the most efficient method.

Let us try the same problem as an arrangement. This gives us

$$\begin{aligned}\left(1+\frac{x^2}{2}+\frac{x^4}{4!}+\cdots\right)\left(x+\frac{x^3}{3!}+\frac{x^5}{5!}+\cdots\right) &= \cosh(x)\sinh(x)\\ &= \frac{e^x+e^{-x}}{2}\frac{e^x-e^{-x}}{2}\\ &= \frac{e^{2x}-e^{-2x}}{4}\\ &= \frac{1}{2}\sinh(2x)\\ &= \frac{1}{2}\sum_{n=0}^{\infty}\frac{(2x)^{2n+1}}{(2n+1)!}\\ &= \frac{1}{2}\sum_{n=0}^{\infty}2^{2n+1}\frac{x^{2n+1}}{(2n+1)!}\\ &= \sum_{n=0}^{\infty}2^{2n}\frac{x^{2n+1}}{(2n+1)!}.\end{aligned}$$

Again we see that only an odd number of objects, $2n+1$, can be arranged, now in 2^{2n} ways. Let us try to justify this as well. If $2r+1$ of the objects come from $\{1,3,5,...,2n+1\}$ then there are $\binom{2n+1}{2r+1}$ ways of arranging these. The remaining identical objects are then placed in the remaining slots. Thus we have

$$\sum_{r=0}^{n}\binom{2n+1}{2r+1}$$

ways of arranging the objects. This can be simplified using a binomial identity. Keep in mind that we are selecting all of the subsets with an odd number of elements. Since precisely half of all subsets have an odd number of elements we always have

$$\binom{n}{1}+\binom{n}{3}+\binom{n}{5}+\cdots=\frac{1}{2}2^n=2^{n-1}.$$

In particular,

$$\sum_{r=0}^{n}\binom{2n+1}{2r+1}=2^{2n}.$$

Next let us try to arrange n objects such that we use at least one of each type. The exponential generating function for each type is given by

$$x+\frac{x^2}{2}+\frac{x^3}{6}+\cdots=e^x-1$$

and the total generating function is

$$\begin{aligned}
\left(e^x-1\right)^k &= \sum_{r=0}^{k}\binom{k}{r}e^{rx}\left(-1\right)^{k-r} \\
&= \sum_{r=0}^{k}\left(-1\right)^{k-r}\binom{k}{r}\left(\sum_{n=0}^{\infty}\frac{\left(rx\right)^n}{n!}\right) \\
&= \sum_{r=0}^{k}\sum_{r=0}^{\infty}\left(-1\right)^{k-r}\binom{k}{r}r^n\frac{x^n}{n!} \\
&= \sum_{n=0}^{\infty}\left(\sum_{r=0}^{k}\left(-1\right)^{k-r}\binom{k}{r}r^n\right)\frac{x^n}{n!} \\
&= \sum_{n=0}^{\infty}k!\left\{ {n \atop k}\right\}\frac{x^n}{n!}
\end{aligned}$$

Thus there are

$$\sum_{r=0}^{k}\left(-1\right)^{k-r}\binom{k}{r}r^n=\sum_{n=0}^{\infty}k!\left\{ {n \atop k}\right\}$$

ways of arranging n objects if we use at least one of each type. The number $\left\{ {n \atop k}\right\}$ is called the Sterling number of the second kind. It counts the number of ways of partitioning a set of n elements into k nonempty subsets. In our case the k types are ordered. Since there are $k!$ ways of doing this we see that $k!\left\{ {n \atop k}\right\}$ is the number of ways of arranging n objects if all k types are used. The formula

$$\begin{aligned}
k!\left\{ {n \atop k}\right\} &= \sum_{r=0}^{k}\left(-1\right)^{k-r}\binom{k}{r}r^n \\
&= \sum_{r=0}^{k}\left(-1\right)^{k-r}\binom{k}{k-r}r^n \\
&= k^n-\binom{k}{1}\left(k-1\right)^n+\binom{k}{2}\left(k-2\right)^n-\cdots
\end{aligned}$$

can also be obtained by an inclusion/exclusion argument. Simply let X be the set of all possible arrangements of n objects of k types and a_i the condition that type i is not used. Then we are trying to calculate

$$N\left(\bar{a}_1\cdots\bar{a}_k\right)=N\left(X\right)-\sum N\left(a_i\right)+\sum N\left(a_ia_j\right)-\cdots$$

Next we indicate how to use generating functions to find the number of solutions to integer equations

$$m_1x_1+m_2x_2+\cdots+m_kx_k=n.$$

This is equivalent to combining n objects of k types with the condition that the i^{th} type is represented by a number of objects divisible by m_i. As an example

let us see how many ways there are of exchanging a one dollar bill for pennies, nickels, dimes and quarters. This corresponds to finding solutions to

$$x_p + 5x_n + 10x_d + 25x_q = 100.$$

The generating functions for the types are

$$\begin{aligned} 1 + x + x^2 + \cdots &= \frac{1}{1-x}, \\ 1 + x^5 + x^{10} + \cdots &= \frac{1}{1-x^5}, \\ 1 + x^{10} + x^{20} + \cdots &= \frac{1}{1-x^{10}}, \\ 1 + x^{25} + x^{50} + \cdots &= \frac{1}{1-x^{25}}. \end{aligned}$$

Combining these gives

$$\frac{1}{1-x}\frac{1}{1-x^5}\frac{1}{1-x^{10}}\frac{1}{1-x^{25}}$$

we now have to find the coefficient in front of x^{100} in order to solve our problem. It doesn't seem feasible to do this by hand. However, quite a number of computer programs can handle this by simply finding the Taylor polynomial of degree 100 for the function. This is not necessarily done by computing the 100^{th} derivative though. The most efficient method might be to calculate out the product.

2.4 Indistinguishable Types

Sometimes types come without any names or indices. For instance if we consider partitions of integers

$$\begin{aligned} 4 &= 1+1+1+1 \\ &= 2+2 \\ &= 1+3 \\ &= 1+1+2 \end{aligned}$$

The objects are 1s and they are grouped as follows

$$\begin{aligned} 1+1+1+1 &= (1)+(1)+(1)+(1), \\ 2+2 &= (1+1)+(1+1), \\ 1+3 &= (1)+(1+1+1) \\ 1+1+2 &= (1)+(1)+(1+1) \\ 4 &= (1+1+1+1). \end{aligned}$$

Unlike the equations we considered above, there is no reason to attach any special index to the types. In this situation we recreate indexed types

$$\begin{aligned} A_1 &= \{0,1,2,3,...\} \\ A_2 &= \{0,2,4,6,...\} \\ A_3 &= \{0,3,6,9,...\} \\ &... \end{aligned}$$

Thus A_1 keeps track of the number of 1s in the partition, A_2 the number of 2s, etc. Therefore, the problem is equivalent to finding integer solutions to

$$x_1 + 2x_2 + 3x_3 + \cdots = n$$

There appear to be an unlimited number of types, but for each n we obviously don't need the types that are larger than n. Thus we have reinterpreted the problem as one of the problems we know how to set up. The generating function is

$$\frac{1}{1-x}\frac{1}{1-x^2}\frac{1}{1-x^3}\cdots$$

where we can terminate the product at n if we are looking for the coefficient in front of x^n.

2.5 Generating Functions for Multiple Groups

Despite statements to the contrary in the textbook it is also possible to create generating functions that calculate how one can combine/arrange different types into several groups. Specifically we want to calculate

$$\begin{bmatrix} A_1, ..., A_k \\ \infty_1, ..., \infty_l \end{bmatrix}_n$$

Note that

$$\begin{bmatrix} A_1, ..., A_k \\ \infty \end{bmatrix}_n = \begin{bmatrix} A_1, ..., A_k \\ n \end{bmatrix}_n$$

so we handled the case were $l = 1$ above.

First we attack the recurrence:

$$\begin{bmatrix} A_1, ..., A_k \\ \infty_1, ..., \infty_l \end{bmatrix}_n = \sum_{n_1+\cdots+n_k=n} \begin{bmatrix} A_1 \\ \infty_1, ..., \infty_l \end{bmatrix}_{n_1} \cdots \begin{bmatrix} A_k \\ \infty_1, ..., \infty_l \end{bmatrix}_{n_k}$$

This is easy to justify as any selection of n objects consists of selecting n_1 objects from A_1, n_2 from A_2 etc and then adding up all possibilities for $n_1, .., n_k$ keeping in mind that $n = n_1 + \cdots + n_k$. By symmetry this looks like a combination with unlimited replacement.

The generating function is

$$\sum_{n=0}^{\infty} \begin{bmatrix} A_1, ..., A_k \\ \infty_1, ..., \infty_l \end{bmatrix}_n x^n$$

and for each type

$$\sum_{n=0}^{\infty} \left[\begin{matrix} A_i \\ \infty_1, ..., \infty_l \end{matrix} \right]_n x^n$$

where

$$\left[\begin{matrix} A_i \\ \infty_1, ..., \infty_l \end{matrix} \right]_n = \left[\begin{matrix} A_i \\ \infty_1, ..., \infty_l \end{matrix} \right]_n = \begin{cases} \binom{l}{n} & \text{if } n \in A_i \\ 0 & \text{if } n \notin A_i \end{cases}$$

The recurrence then tells us that

$$\begin{aligned} \sum_{n=0}^{\infty} \left[\begin{matrix} A_1, ..., A_k \\ \infty_1, ..., \infty_l \end{matrix} \right]_n x^n &= \left(\sum_{n=0}^{\infty} \left[\begin{matrix} A_1 \\ \infty_1, ..., \infty_l \end{matrix} \right]_n x^n \right) \cdots \left(\sum_{n_k=0}^{\infty} \left[\begin{matrix} A_k \\ \infty_1, ..., \infty_l \end{matrix} \right]_n x^n \right) \\ &= \left(\sum_{n \in A_1} \binom{l}{n} x^n \right) \cdots \left(\sum_{n \in A_k} \binom{l}{n} x^n \right) \end{aligned}$$

An example of this of this kind of problem would be to select wines of several types $A_1, .., A_k$ and then distribute them to stores. Generally they'd be grouped in 12 bottle cases and otherwise come in different quantities of $i_1, ..., i_k$ cases for each type. For simplicity let us assume that quantities are unlimited for each type. This is not such a terrible assumption if we are only distributing a small number of cases compared to the supplies. The generating function for distributing n cases is then given by

$$\begin{aligned} \sum_{n=0}^{\infty} \left[\begin{matrix} A_1, ..., A_k \\ \infty_1, ..., \infty_l \end{matrix} \right]_n x^n &= \left(\sum_{n=0}^{\infty} \binom{l}{n} x^n \right)^k \\ &= \left((1-x)^{-l} \right)^k \\ &= (1-x)^{-l \cdot k} \\ &= \sum_{n=0}^{\infty} \binom{l \cdot k}{n} x^n. \end{aligned}$$

This can also be explained by a combinatorial argument. Namely we define $k \cdot l$ types consisting of pairs of a type of wine and a store. Thus we are performing a combination with replacement of n cases into these pairs.

Chapter 3

Recurrence Relations

The types of problems we are going to solve using recurrence will look like arrangements. However, one of the interesting features is that the number of objects used for an arrangement is not fixed. What is fixed is the about of space that the objects occupy. This means that we will be concerned with problems where we have types and the objects come in different sizes as well. We will restrict attention to things that can be interpreted as being one dimensional. Such as placing paving stones, tiles, beads, etc in a line or circle.

3.1 Combinatorial Interpretations

We are going to study two types of *linear homogeneous recurrences*. The *monochromatic* version comes from a recurrence

$$\begin{aligned} a_n &= c_1 a_{n-1} + \cdots + c_k a_{n-k} \\ &= \sum_{i=1}^{k} c_i a_{n-i} \end{aligned}$$

where c_i is either 1 or 0. The Fibonacci sequence is of this type as are many other recurrences. The interpretation is that we pave a sidewalk with paving stones that come in sizes that are a multiple of 1. The coefficient $c_i = 1$ if we are allowed to use stones of length i.

The no consecutive x problem is of this type as we can use the two paving stones xo and o to generate strings without consecutive xs. However, the initial values of a_1 and a_2 depend on the exact interpretation of the problem. With the paving prblem we have $a_1 = 1$ and $a_2 = 2$, while with the no consecutive xs we have $a_1 = 2$ and $a_2 = 3$. Thus one sequence is shifted from the other.

The *polychromatic* version looks the same

$$\begin{aligned} a_n &= c_1 a_{n-1} + \cdots + c_k a_{n-k} \\ &= \sum_{i=1}^{k} c_i a_{n-i} \end{aligned}$$

except c_i is now a nonnegative integer. This time we can use an arts and craft interpretation. We have beads that come in integer sizes and beads of size i come in c_i different varieties. These beads are placed on a string that can be made into a necklace or bracelet.

The *initial conditions* are the first k values $a_1, ..., a_k$. These have to be determined from each individual problem. Let us focus on $k = 2$ and assume that we are placing beads on a string from right to left. Clearly $a_1 = c_1$ as there are c_1 beads of size 1 to fill a space of size 1. To fill a space of size 2 we can use two beads of size 1 or one bead of size 2. Thus $a_2 = c_1^2 + c_2$. It is often convenient to know the value of a_0. It has no obvious meaning but we can nevertheless assign a value by using that

$$a_2 = c_1 a_1 + c_2 a_0,$$

so

$$\begin{aligned} a_0 &= \frac{a_2 - c_1 a_1}{c_2} \\ &= \frac{c_1^2 + c_2 - c_1^2}{c_2} \\ &= 1 \end{aligned}$$

Clearly the two initial conditions $a_0 = 1$ and $a_1 = c_1$ are easier to remember and work with.

Rather than placing our objects on a string from left to right we can also place them in a circle, e.g., making a bracelet rather than a necklace. The bracelet doesn't seem to have a fixed point. The simplest way around this is to assume that the bracelet is fixed by hanging from a peg. If we take the bracelet off the peg and rotate it it'll give us a new configuration. To get a recurrence going we delete the bead that is opposite the peg, or just to the left of the point opposite the peg. If the peg is at 12 o'clock then we remove the bead that covers 6 o'clock or is just to the left of 6 o'clock. If this bead has size i we get a string of size $n - i$ that can be glued back into a bracelet. This analysis shows that bracelets satisfy the same recurrence as the necklaces:

$$\begin{aligned} a_n &= c_1 a_{n-1} + \cdots + c_k a_{n-k} \\ &= \sum_{i=1}^{k} c_i a_{n-i} \end{aligned}$$

The difference comes when determining the initial values. Again let us just consider the case where $k = 2$. One bead bracelets are made by just one bead

of size 1, so $a_1 = c_1$. Two bead bracelets are made either from 2 beads of size 1 (c_1^2 possibilities) or from 1 bead of size 2. This last case is a bit tricky. There are c_2 of the size 2 beads, but each bead can make a necklace in two different ways. We can make the bead close up at 12 or 6 o'clock. Thus there are $2c_2$ ways of making bracelets with beads of size 2.

The initial conditions

$$\begin{aligned} a_1 &= c_1, \\ a_2 &= c_1^2 + 2c_2 \end{aligned}$$

now give us

$$\begin{aligned} a_0 &= \frac{a_2 - c_1 a_1}{c_2} \\ &= \frac{c_1^2 + 2c_2 - c_1^2}{c_2} \\ &= 2 \end{aligned}$$

So necklaces and bracelets have the same initial value $a_1 = c_1$, but different a_0.

A somewhat different recurrence is given by

$$D_n = (n-1)\left(D_{n-1} + D_{n-2}\right)$$

this comes from the derangement problem of counting the number of permutations without fixed points. That is arrangements of $1, ..., n$ where 1 is never in 1st place, 2 never in 2nd, etc. We can justify this recurrence as follows: 1 goes to the i^{th} location where $i = 2, ..., n$. There are $n-1$ choices for this. Now i can go to the 1^{st} location in which case the remaining $n-2$ elements get deranged. So there are $(n-1)\,D_{n-2}$ possibilities for having 1 go to i and i to 1. Next assume that 1 goes to i and that i doesn't go to 1. Since the i^{th} location has been occupied by 1 we can ignore it and then rename the 1^{st} spot as i. Since i was not sent to 1 we are performing a derangement of $n-1$ elements. Thus there are $(n-1)\,D_{n-1}$ possibilities for having 1 go to i and not having i go to 1.

As it is impossible to derange 1 object we have $D_1 = 0$ and there is only one way of deranging 2 objects so $D_2 = 1$. The recurrence then indicates that $D_0 = 1$.

If we rewrite this recurrence as

$$\begin{aligned} D_n - nD_{n-1} &= -D_{n-1} + (n-1)\,D_{n-2} \\ &= -\left(D_{n-1} - (n-1)\,D_{n-2}\right) \end{aligned}$$

we see that

$$D_n - nD_{n-1} = (-1)^n$$

or

$$D_n = nD_{n-1} + (-1)^n$$

Thus we have reduced a natural second order recurrence to a first order recurrence. However, it is not clear that there is a simple combinatorial proof of the first order recurrence.

3.2 Solving Recurrences

When solving a general recurrence there are several strategies.

3.2.1 Reductionist Approach

The *reductionist* approach is the one most often used. It tries to find solutions by solving simpler recurrences and then finding the original solution by forming linear combinations of the simpler solutions. As the simplest recurrence is $a_n = c_1 a_{n-1}$ and this recurrences has $a_n = c_1^n$ as a solution we usually look for solutions of the form $a_n = c^n$. Testing when such a sequence is a solution comes down to checking

$$c^n = c_1 c^{n-1} + \cdots + c_k c^{n-k}$$

If we factor out c^{n-k} this means that we are trying to solve

$$c^k = c_1 c^{k-1} + \cdots + c_{k-1} c + c_k.$$

In other words we find solutions $a_n = c^n$ where c is a root of the characteristic equation. It is worth pointing out that the roots are rarely nice numbers. In fact the only rational c that can be roots are integers that divide c_k.

3.2.2 Shift Approach

A somewhat different strategy comes about by observing that if we have a solution a_n then we can create other solutions by shifting the index

$$a'_n = a_{n+l}$$

where l is an integer that can be both positive and negative.

If we consider the Fibonacci type recurrence

$$a_n = a_{n-1} + a_{n-2}$$

the reductionist approach asks us to find c that solve

$$\begin{aligned} c^2 &= c + 1, \\ c &= \frac{1 \pm \sqrt{5}}{2} \end{aligned}$$

Certainly this allows us to find all solutions, but not with the nicest possible expressions. If instead we agree that the classical Fibonacci sequence that has

the initial values $f_0 = 1$ and $f_1 = 1$ is good enough, then we can quickly create a new solution by shifting

$$\begin{aligned} a_0 &= f_1 = 1, \\ a_1 &= f_2 = 2, \\ a_n &= f_{n+1} \end{aligned}$$

All solutions are now linear combinations of these two solutions. As an examples let us find the Lucas numbers. They form the sequence that comes from the same recurrence but with the initial numbers $L_0 = 2$ and $L_1 = 1$ that comes form considering bracelets. We have to solve

$$\begin{aligned} 2 &= L_0 = \alpha f_0 + \beta a_0 = \alpha + \beta, \\ 1 &= L_1 = \alpha f_1 + \beta a_1 = \alpha + 2\beta \end{aligned}$$

This gives us

$$\begin{aligned} \beta &= -1, \\ \alpha &= 3 \end{aligned}$$

Thus

$$L_n = 3f_n - f_{n+1}$$

But this is not the only possibility. if we allow for different shifts we see that

$$\begin{aligned} L_n &= 3f_n - f_{n+1} \\ &= 3f_n - f_n - f_{n-1} \\ &= 2f_n - f_{n-1} \\ &= f_n + f_{n-2}. \end{aligned}$$

Thus we could have shifted the Fibonacci sequence back twice and gotten a much nicer formula. The formula $L_n = f_n + f_{n-2}$ has a simple combinatorial explanation. When we break the necklace two things can happen. Either no bead covers 6 o'clock and so we get a string of length n, or a bead of size 2 covers 6 o'clock in which case we get a string of length $n - 2$.

More generally, if the necklace solution to a polychromatic recurrence

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

is given by a_n and the bracelet solution by b_n, then

$$b_n = a_n + c_2 a_{n-2}.$$

3.2.3 Generating Function Approach

We gather the solution a_n to a recurrence in a generating function

$$g(x) = \sum_{n=0}^{\infty} a_n x^n$$

In order to solve

$$a_{n+2} = c_1 a_{n+1} + c_2 a_n$$

we multiply both sides by x^{n+2} and add up over all $n = 0, 1, ...$ to get

$$\sum_{n=0}^{\infty} a_{n+2} x^{n+2} = c_1 \sum_{n=0}^{\infty} a_{n+1} x^{n+2} + c_2 \sum_{n=0}^{\infty} a_n x^{n+2}$$

We then observe that

$$\begin{aligned}
\sum_{n=0}^{\infty} a_{n+2} x^{n+2} &= a_2 x^2 + a_3 x^3 + \cdots \\
&= g(x) - a_0 - a_1 x, \\
\sum_{n=0}^{\infty} a_{n+1} x^{n+2} &= x \sum_{n=0}^{\infty} a_{n+1} x^{n+1} \\
&= x\left(g(x) - a_0\right), \\
\sum_{n=0}^{\infty} a_n x^{n+2} &= x^2 \sum_{n=0}^{\infty} a_n x^n \\
&= g(x)
\end{aligned}$$

giving us the algebraic equation for $g(x)$

$$g(x) - a_0 - a_1 x = c_1 x\left(g(x) - a_0\right) + c_2 x^2 g(x)$$

Isolating $g(x)$ then tells us that

$$g(x) = \frac{a_0 + (a_1 - a_0 c_1) x}{1 - c_1 x - c_2 x^2}$$

After using partial fractions we can then expand the RHS in a powerseries and find the coefficients a_n.

It is interesting to note that the generating function for a linear recurrence is always a rational function where the denominator depends only on the recurrence relation in the form just described. The numerator has degree less than the order of the recurrence and depends on the initial values of the specific sequence we wish to find.

Conversely any reasonable rational function comes from a recurrence.

3.2.4 Exponential Generating Function Approach

Finally we show a far more pleasing approach that ties in with your knowledge of differential equations. While it doesn't seem to give us a new way of solving recurrences it combines all of the above approaches into one.

This time we use the exponential generating function

$$g(x) = \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}$$

This makes quite a bit of sense as we are considering an arrangement problem anyway. Also the relationship between g and the coefficients is a bit more familiar as it is given by Taylor's formula

$$a_n = \frac{d^k g}{dx^k}(0).$$

Next we note what happens when g is differentiated

$$\begin{aligned} g'(x) &= \sum_{n=1}^{\infty} a_n n \frac{x^{n-1}}{n!} \\ &= \sum_{n=1}^{\infty} a_n \frac{x^{n-1}}{(n-1)!} \\ &= \sum_{n=0}^{\infty} a_{n+1} \frac{x^n}{n!} \end{aligned}$$

Thus shifting the index corresponds to differentiating g. If we multiply the second order recurrence

$$a_{n+2} = c_1 a_{n+1} + c_2 a_n$$

by $x^n/n!$ and add we get

$$\sum_{n=0}^{\infty} a_{n+2} \frac{x^n}{n!} = c_1 \sum_{n=0}^{\infty} a_{n+1} \frac{x^n}{n!} + c_2 \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}$$

or the second order linear differential equation

$$g'' = c_1 g' + c_2 g.$$

Such equations are solved using the reductionist approach, namely, we seek g that solve first order equations

$$g' = cg$$

Since that equation is solved by $g = \exp(ct)$ we see that the second order equation becomes

$$c^2 \exp(ct) = c_1 c \exp(ct) = c_2 \exp(ct)$$

After eliminating $\exp(ct)$ this is the characteristic equation

$$c^2 = c_1 c + c_2$$

form above.

The advantage of exponential generating functions becomes more obvious when we are studying slightly more complicated recurrences. For instance the derangement problem leads to the recurrence

$$D_{n+1} = (n+1) D_n + (-1)^{n+1}$$

This is a linear first order inhomogeneous recurrence. The problem lies in having a coefficient that isn't fixed. The initial condition is $D_0 = 1$. Transforming it to exponential generating functions

$$g(x) = \sum_{n=0}^{\infty} D_n \frac{x^n}{n!}$$

now gives

$$\sum_{n=0}^{\infty} D_{n+1} \frac{x^n}{n!} = \sum_{n=0}^{\infty} (n+1) D_n \frac{x^n}{n!} + \sum_{n=0}^{\infty} (-1)^{n+1} \frac{x^n}{n!}$$

The LHS is simply g'. The RHS can be reduced as follows

$$\begin{aligned}
\sum_{n=0}^{\infty} (n+1) D_n \frac{x^n}{n!} + \sum_{n=0}^{\infty} (-1)^{n+1} \frac{x^n}{n!} &= \left(\sum_{n=0}^{\infty} D_n \frac{x^{n+1}}{n!} \right)' - \exp(-x) \\
&= \left(x \sum_{n=0}^{\infty} D_n \frac{x^n}{n!} \right)' - \exp(-x) \\
&= (x g(x))' - \exp(-x) \\
&= g + x g' - \exp(-x)
\end{aligned}$$

Thus

$$g' = g + x g' - \exp(-x)$$

or

$$(x-1) g' + g = \exp(-x)$$

This is easily solved by noting that the LHS

$$(x-1) g' + g = ((x-1) g)'$$

and the initial condition is

$$D_0 = g(0) = 1.$$

Thus

$$(x-1) g(x) = -\exp(-x)$$

or

$$g(x) = \frac{\exp(-x)}{1-x}.$$

3.3 Nonlinear Recurrences

These are quite common and need to be solved using a variety of tricks.